02188.64-02

EDA Build 2
Final Report

Prepared for:
NASA Ames Research Center
Moffett Field, CA

under

CONTRACT:	NAS2-98005
SUBCONTRACT:	99-0249
TASK ORDER NO:	64
SEAGULL PROJECT:	C188.64

Prepared by:
Susan Dorsky
David Signor

Seagull Technology, Inc.
16400 Lark Avenue
Los Gatos, CA 95032-2547

March 31, 2002

Acknowledgment

Seagull Technology Inc., as a subcontractor to Titan Systems/SRC division, continues to make great strides to bring CAST functionality (conflict free advisories for metered aircraft) into the CTAS baseline. The authors appreciate the extensive discussions, technical direction and program management of Rich Coppenbarger within the En Route Operations branch of NASA Ames Research Center. The authors also wish to thank Matt Jardin, NASA aerospace engineer, for his work with Path Stretch and members of the CTAS software group for their help with understanding the CTAS code and group procedures. We would especially like to recognize Karen Cate, Michelle Eshow, John Robinson, Dr. Danny Chiu and Rey Salcido for their assistance. Seagull also wants to recognize Robert Vivona and Husni Idris of Titan/SRC Division without whose help there would be no metered arrivals and the RAPS architecture would still be a mystery.

Seagull software engineer Erin Wallace was a major contributor to Build 2 development by performing the PGUI software modifications and making initial progress into the implementation of the Slattery algorithm. Anna Dabrowski contributed to this document by learning to run PAS, setting up initial scenarios and creating screen captures. Liang Chen has brought his extensive CTAS knowledge to the project and is continuing with the conflict resolution implementation. Ari Stassart has contributed to the project by sharing his CTAS knowledge with regard to how D2 works in the en-route airspace. The authors would also like to express their gratitude to Dave Schleicher for his system engineering input and support and his review of this document.

Table of Contents

Acronyms

AATT	Advanced Air Transportation Technology
ACID	Aircraft Identification
ACL	Active Conflict List
ATM	Air Traffic Management
ATR	Advisory Time Range
ATSP	Air Traffic Service Provider
CAS	Calibrated Airspeed
CAST	Center Advisory and Spacing Tool
CD&R	Conflict Detection and Resolution
CM	Communications Manager
CPU	Central Processor Unit
CTAS	Center TRACON Automation system
D2	Direct-To
DAG-TM	Distributed Air Ground Traffic Management
DOF	Degree of Freedom
DP	Dynamic Planner
DST	Decision Support Tool
EDA	En-Route Descent Advisor
GUI	Graphical User Interface
ISM	Input Source Manager
MF	Metering Fix
MHz	Mega Hertz
NASA	National Aeronautics and Space Administration
PAS	Pseudo Aircraft Systems
PAZ	Protected Airspace Zone
PFS_C	Profile Selector-Center process
PGUI	Planview Graphical User Interface
RA	Route Analyzer process
RCL	Resolution Conflict List
SSS	System Sub-System Specification
STA	Scheduled Time of Arrival
STAR	Standard Terminal Arrival Route
TMA	Traffic Management Advisor
TN	Trajectory Negotiation
TOD	Top of Descent
TS	Trajectory Synthesizer

Terms

Direct To	CTAS tool that provides advisories for departures or overflights to fly directly to another waypoint (other than next) on their flight plan.
Degree of Freedom	A variable that may be adjusted to achieve a desired result. For EDA, the DOFs are cruise speed, descent speed and altitude. Flight path may also be considered a DOF but is handled differently.

RAPS	Common libraries for the RA and PFS processes and their interface with the TS process. This term also refers to the design architecture that handles these processes.
Stream Class	Aircraft that have the same engine type and are going to the same airport through the same metering fix.

1 Introduction

To support NASA's Distributed Air Ground Traffic Management (DAG-TM) Concept, the Advanced Air Transportation Technologies (AATT) project is pursuing the development and validation of Air Traffic Management (ATM) Decision Support Tool (DST) technologies and procedures. A major element of DAG is Trajectory Negotiation (TN). The goal of TN, is to improve user flexibility, throughput, efficiency and controller productivity within flow-constrained en route airspace. The problem to be researched involves the transition from "free flight" to high-density terminal airspace impacted by arrival metering delays.

To enable TN, NASA must first develop and evaluate decision support technology, and supporting procedures for the air traffic service provider (ATSP). A fundamental requirement for TN is the ability of controllers to manage traffic with a "trajectory orientation" rather than the current-day procedures which are "sector oriented"[AATTNRA99]. A trajectory orientation uses strategic planning to minimize downstream problems. Such procedures will require advanced decision support capabilities. The ATSP automation will be part of the Center TRACON Automation System (CTAS) baseline.

The en-route element of CTAS is the En Route Descent Advisor (EDA). EDA assists the controller in planning fuel-efficient, conflict-free trajectories with the goal of reducing deviations from the user's preferred trajectory. The EDA algorithms generate advisories for conflict resolution and conformance with traffic-management restrictions such as crossing altitude/speed, and flow-rate. Efficient conformance with flow-rate restrictions is important because these restrictions are a major cause of user deviations that are often required to reduce overloading the airspace and/or airport capacity. The flow-conformance automation also provides conflict detection and resolution (CD&R) functions with accurate models of intent, thus reducing the false alarm and missed-alert rates associated with flow-constrained operations. A fundamental contribution of the EDA concept is the integration of flow-rate conformance and CD&R to deliver total system performance that is greater than the sum of the contributions from each part [GV99], [SG94].

EDA will service en-route airspace including aircraft in the climb, cruise, and descent phases of flight. By providing decision support to enable controllers to operate with a trajectory orientation, EDA establishes a foundation for the following:

1. Reduction in route and crossing restrictions
2. More favorable distribution and reduction in sector workload
3. Reduction in ATC "interruptions" due to corrective clearances
4. Improved flow-rate conformance efficiency
5. Increased user preferred trajectories and flexibility through:
   - Distributed decision making (e.g., trajectory negotiation);

- Distributed responsibility (e.g., separation and flow-conformance assurance).

1.1 References

[AATTNRA99] AATT NRA TO-34 SOW, April 1999

[GV99] Green, S., and Vivona, R., *En route Descent Advisor Concept Definition, AATT Milestone 5.10 report*, September, 1999.

[SG94] Slattery, R., Green, S., *Conflict-Free Trajectory Planning for Air Traffic Control Automation*, NASA Technical Memorandum 108790, January 1994.

[SSS] *En Route Descent Advisor (EDA) Build 3 System Specification*, prepared by System Resources Corporation for NASA Contract: RTO-45: NAS2-98005, December 29, 2000.

[MP00] Peters, M., *En Route Descent Advisor Build 2 Functional Requirements*, Final Report (00188.46-02) prepared for NASA Contract: RTO-46: NAS2-98005, December 29, 2000.

[IVM01] Idris, H., Vivona, R., McDonald, J., *Design Document for the Implementation of Speed-Mode Meet-Time in RAPS*, prepared for NASA Contract: RTO-63: NAS2-98805.

[DWS01] Dorsky, S., Wallace, E., Schleicher, D., *Path Stretching in CAST and CTAS*, Seagull Technology Inc. internal report, June 2001.

[DWS02] Dorsky, S. Wallace, E., Schleicher, D., *EDA Build 2 Conflict Resolution Design Document*, Seagull Technology Inc., prepared for NASA Contract: RTO-64:NAS2-98005, September 24, 2001.

1.2 Background

EDA development is planned as a series of builds. Builds 1 and 2 provide research tools that support near-term part-task assessments of trajectory-based operations without regard to sector and facility jurisdictional structures and associated inter-sector coordination (i.e., “single-sector” tools). These initial builds introduce and examine CD&R and flow conformance integration. Build-1 was developed to provide a basic miles-in-trail spacing tool capability within the CTAS baseline. (Note: Build 1 is not in the critical path for the other builds.) Build 2 focuses on arrival metering, and represents the implementation of Center Automation and Sequencing Tool (CAST) functionality within the CTAS baseline. The current technical challenge is to advance this multi-build program to successfully develop EDA.

The CAST functionality was investigated and documented at the end of Build 1 in the document [MP00]. At that time it was determined that the CTAS baseline contained some of the functionality already but it needed to be activated and tested within the current environment. The biggest problem was the change in the Trajectory Synthesizer (TS) process. Under the "old" TS, written in C code, and in CAST, the flow-rate conformance advisory (also referred to as meet-time advisory) was calculated within TS. With the "new" TS, written in C++ code, this functionality did not exist. It was determined that NASA would provide this functionality.

To assist NASA in dealing with the new TS, two meet-time modes, Descent Only and Cruise Only, were implemented for NASA by Titan Systems Corporation, SRC Division under the RTO63 contract and are documented in [IVM01]. The meet-time advisories for

these two modes are now calculated in the PFS_C process interacting with the "new" TS. Details on the meet-time functionality of Build 2 are not within the scope of this document.

2 EDA Build 2 Tasks

Table 2-1 shows the tasks that were initially determined to be necessary to implement CAST functionality within the CTAS baseline. After further examination it was determined that displaying the advisories under the aircraft data tag required new software code (noted by the X in the second column and the extra ✔ in the third column of the next to last task.)

Table 2-1 Initial EDA Build 2 Task List

Functional Requirement: EDA Capability	Satisfied by CTAS Baseline	Presumably satisfied in CTAS Baseline but suppressed	Requires new software code
Planview display with routes and aircraft icons	✔		
Standard data tags	✔		
Timeline which can display STAs and ETAs	✔		
Customizable timeline to filter aircraft based on meter fix	✔		
Timeline must display slow and fast ETAs for the selected aircraft	✔		
Ability to manually manipulate STAs from the timeline			✔
Conflict Detection with appropriate graphics	✔		
Variable separation standards for conflict probe	✔		
Conflict pairs displayed in a conflict window	✔		
Conflict point of loss of separation shown on planview display	✔		
Path stretching capability		✔	
Top of Descent Marks		✔	
Provide cruise advisories		✔	
Provide descent advisories		✔	
Provide conflict resolution advisories to a metered arrival			✔
Allow user to control type of advisories to be issued		✔	
Display advisories in an advisory window in proper format		✔	
Display advisories under aircraft data tag		*X*	✔
Independent **–eda** initiation option for the PGUI			✔

The project tasks were then divided into four primary groups – -eda option development, advisory development, TS development, and conflict resolution development. These groups and the tasks they include along with auxiliary tasks are shown in Figure 2-1.

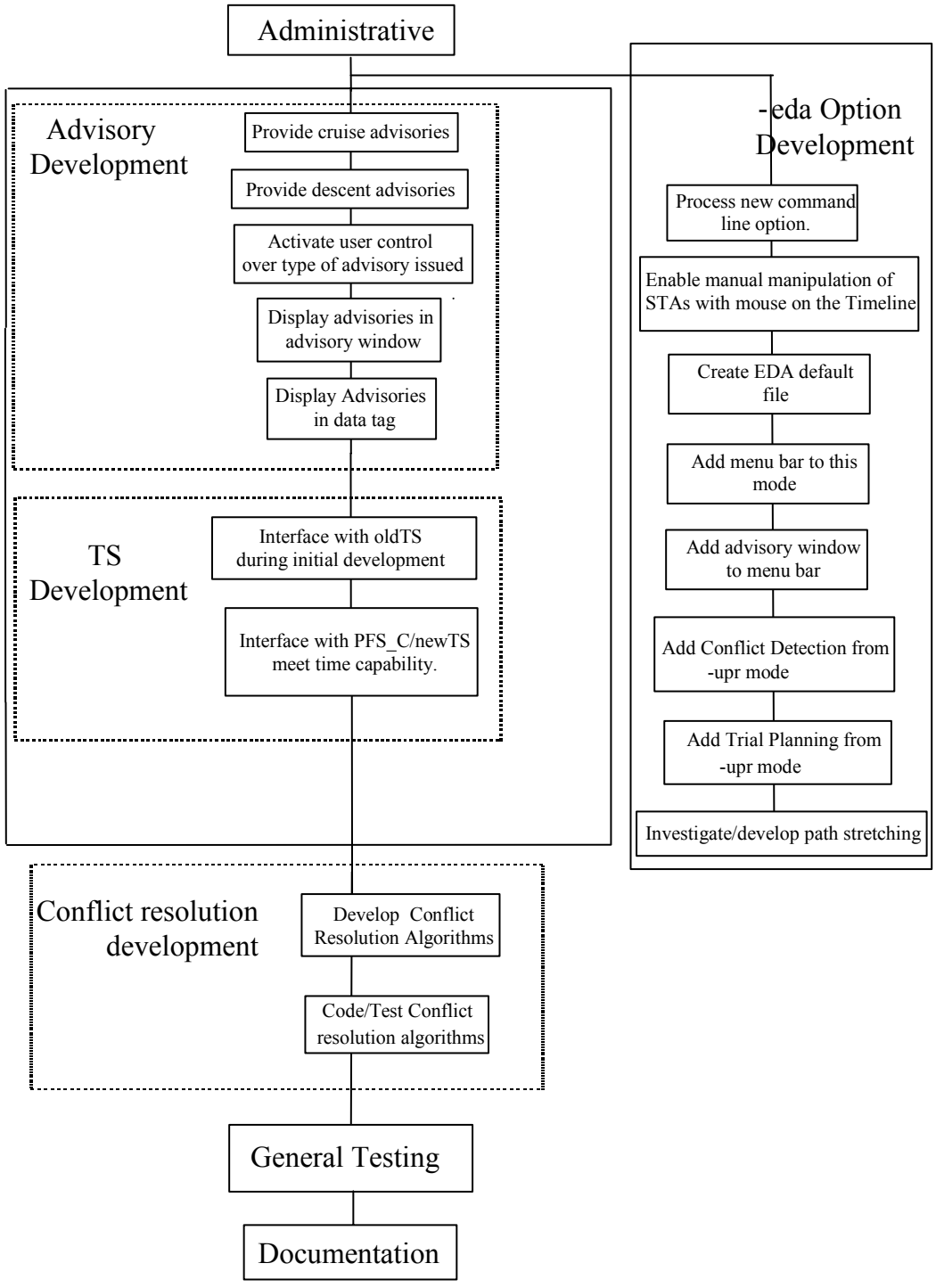


Figure 2-1 EDA Build 2 Task Flow

2.1 –eda Options Development

After much discussion, it was determined that a separate mode for the PGUI process was necessary for the EDA Build 2 implementation. The default PGUI initiation (with no command line mode arguments) was the old DA mode. However, since we would be borrowing some functionality from the –upr mode and adding other functionality, we needed a means to determine when to include an item and when not to. We decided that the addition of the –eda mode would satisfy these requirements. Figure 2-2 shows the PGUI with the exploited/additional/changed items implemented for EDA.

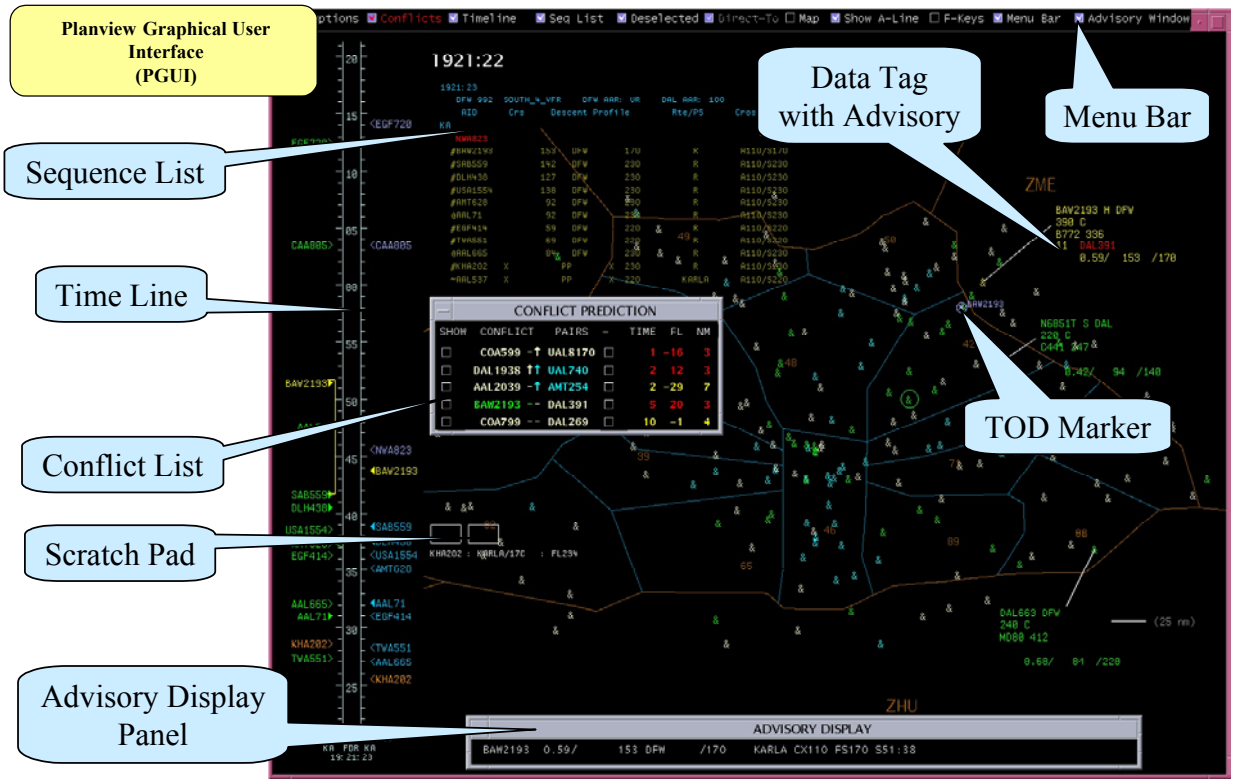


Figure 2-2 PGUI with EDA items highlighted

The items borrowed from the –upr mode are the menu bar and conflict list shown, along with other EDA user interface elements, in Figure 2-2. We also wanted to leave the Trial Planning functionality as it is used in the Direct-To/Trial Planner (D2TP) tool although it is not integrated with EDA at this time. We modified the menu bar by adding a toggle for the advisory window. (See section 2.2 for further information on the advisory window.) We turned off the automatic display of the Trial Plan Accept Panel and only show it when Trial Planning is taking place. We also connected the display of the Flight Plan Panel with the display of the Trial Plan Accept Panel. Figure 2-3 shows the Flight Plan Panel's position without the Trial Plan Accept Panel and Figure 2-4 shows the position when the Trial Plan Accept Panel is displayed.

Figure 2-3 Flight Plan Panel at the bottom of PGUI screen

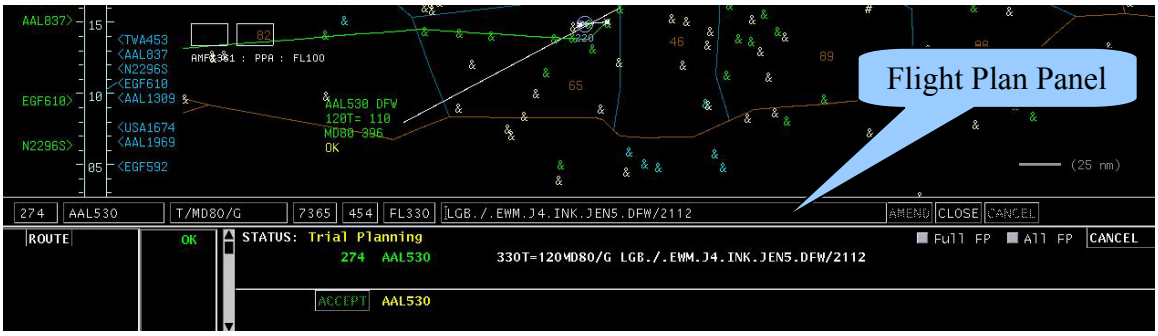


Figure 2-4 Flight Plan Panel on top of Trial Plan Accept Panel

The –eda mode also added the ability to accept trial plans for arrivals. Currently, in Direct To (-upr mode), vectoring and altitude trial plan times are not being calculated for arrivals. As the code currently exists, if these times are not available, the "Accept" button on the Trial Plan Accept panel is not activated. Since we are specifically interested in arrivals, we turned on the Accept button for the EDA case, however, since Trial Planned aircraft are not integrated with meet-time advisories yet, the use of this functionality is not currently recommended.

2.2 Advisory Development

In the CTAS baseline, advisories were being displayed in the Sequence List for aircraft in selected arrival streams (as shown in Figure 2-6) and in an advisory panel for selected aircraft (as shown in Figure 2-7). To place the advisory in the data tag was a requirement in this build. The format used for the data tag was based on the format of the Advisory Panel. As shown in Figure 2-8, the first item is the current cruise speed followed by a '/' character. If there was a cruise advisory, it would follow the '/' and the speed would either be represented by a Mach number or a CAS value. Since the example shown was for a Descent Only advisory, this value is left blank. The second item is the DME distance from the top of descent (TOD) to the destination fix (in this case DFW). The next value is the descent speed. If the Cruise Only advisory mode is used, this will be the current planned descent speed (usually the company profile). If it is an advisory, the Mach/CAS are advisories. An altitude advisory is not shown in the example, but if one were given, the user would see the letters 'FL' followed by an advised flight level. Finally, a Path Stretch advisory would appear at the end, if one were given. This advisory would be the letters 'PS' followed by the distance to the turnback point (or UTC time at the turnback point)/turnback heading. Figure 2-5 is a theoretical example of a complete advisory with cruise and descent speeds, TOD distance, altitude and path stretch.

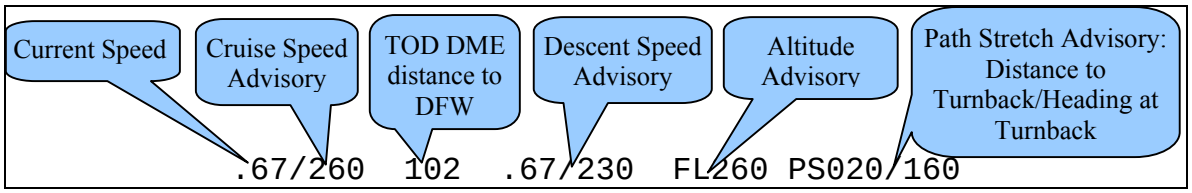


Figure 2-5 Complete Advisory Example

For the default EDA mode, the advisory was placed in the second field of the fifth line of the data tag. As with most CTAS tools, the data tag is completely configurable. Any of the possible values may be placed in either of the 2 fields in any of the 5 available lines. There may also be more than one item in a field and then the items time-share the space according to a pre-set time value. It is presumed that further user evaluation will take place to determine the final format and placement of the advisory information.

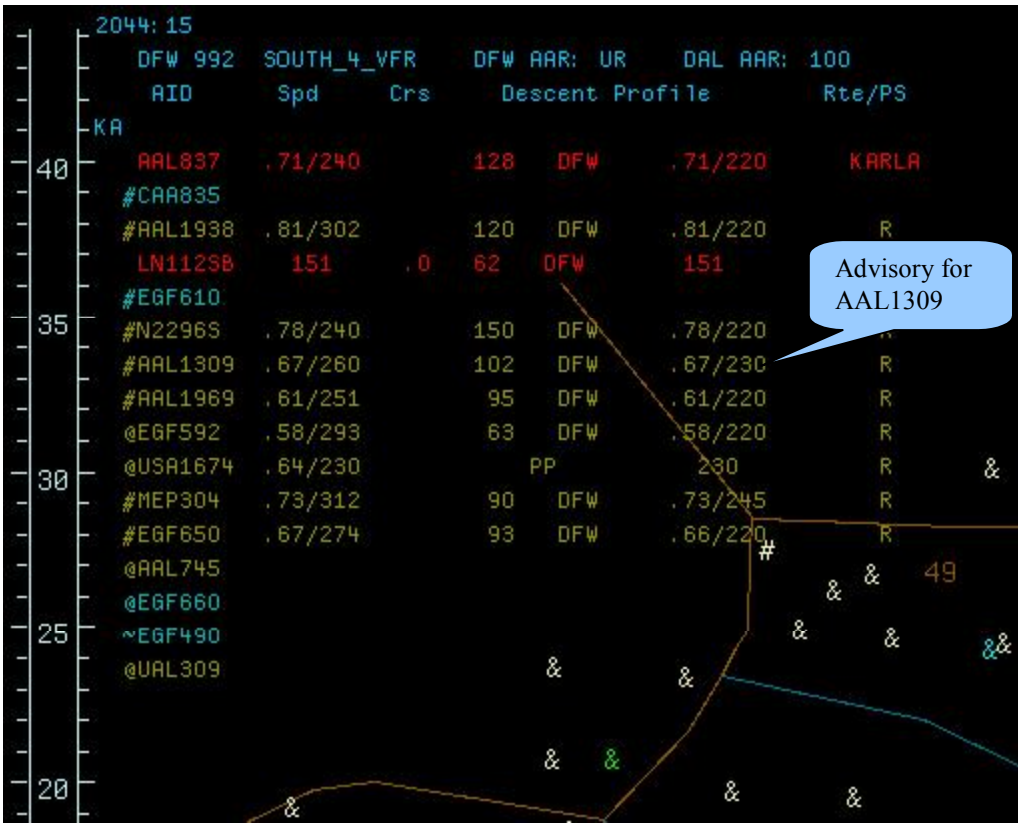


Figure 2-6 Advisories in Sequence List

Figure 2-7 Advisory in Advisory Panel

Figure 2-8 Advisory in the Data Tag

2.3 TS Development

In the original CTAS, the TS process iterated on the trajectory to come up with one that met the given time. When TS was re-written in C++ by the NASA CTAS software group, this functionality was eliminated due to limited time and also to a change in philosophy. The CTAS software group felt that the TS should generate one trajectory given a set of input parameters. The iteration on meet-time has now been moved to the PFS_C process. Briefly, the method works as follows. The RA determines estimated-time-of-arrival (ETA) for the aircraft for various speeds (fastest, slowest, company preferred) to all of the possible runways. Once PFS_C receives these data it then selects an iteration space given certain known parameters such as the aircraft's current speed and altitude and the time it wants to meet. PFS_C picks some speed within this iteration space and TS comes up with the trajectory that uses this new input. If the trajectory still does not meet the time, PFS_C picks a new speed and tries again until a trajectory is found that best meets the time within a given range.

In order to move this iteration to PFS_C, and conform to the RAPS architecture (see Section 2.5 for more information on RAPS), new analysis categories needed to be developed. New categories were needed for each of the possible meet-time modes. These modes are Descent Only, Cruise Only, Cruise then Descent, Cruise equals Descent, Slowest Descent Cruise and Cruise Only no lead distance. The Descent Only and Cruise Only modes were implemented for NASA by Titan Systems Corporation, SRC Division under the RTO63 contract and are documented in [IVM01].

During this work a number of TS interface requirements were specified and submitted to NASA. NASA implemented two of the submitted items, the MAX and MIN flags. These flags are used to request the fastest and slowest speed trajectories for both the descent-only and cruise-only modes.

It was also noted that the top-of-descent (TOD) in the advisory was not calculated correctly by the new TS. A path distance was being used instead of calculating the radial

distance from the TOD to the VOR. The CTAS software group corrected this problem once Seagull showed it to them.

2.4 Path Stretch

Path Stretch, in the Build 2 context, is the ability to change the lateral path of a flight in order to meet a time. Path Stretch would be implemented if the aircraft could not meet the STA with speed and altitude adjustments alone. It was assumed that this capability was still within the CTAS baseline but suppressed. Seagull performed initial testing with the old TS. After exploration, Seagull determined that to display the Path Stretch advisories, two advisory category files from the Denver (DIA) adaptation data had to be borrowed. These files, advisory_categories and advisory_category_definitions, were stored in the adaptation/ZFW_DFW/system directory after they were edited to change the heading NEW_DENVER(DIA) to DALLAS_FT_WORTH. Once these files were in place, it was determined that Path Stretch partially worked with the old TS process as shown in Figure 2-9. Seagull documented how Path Stretch worked with the original DA tool in CAST and the way it was initiated in the current CTAS baseline [DWS01].

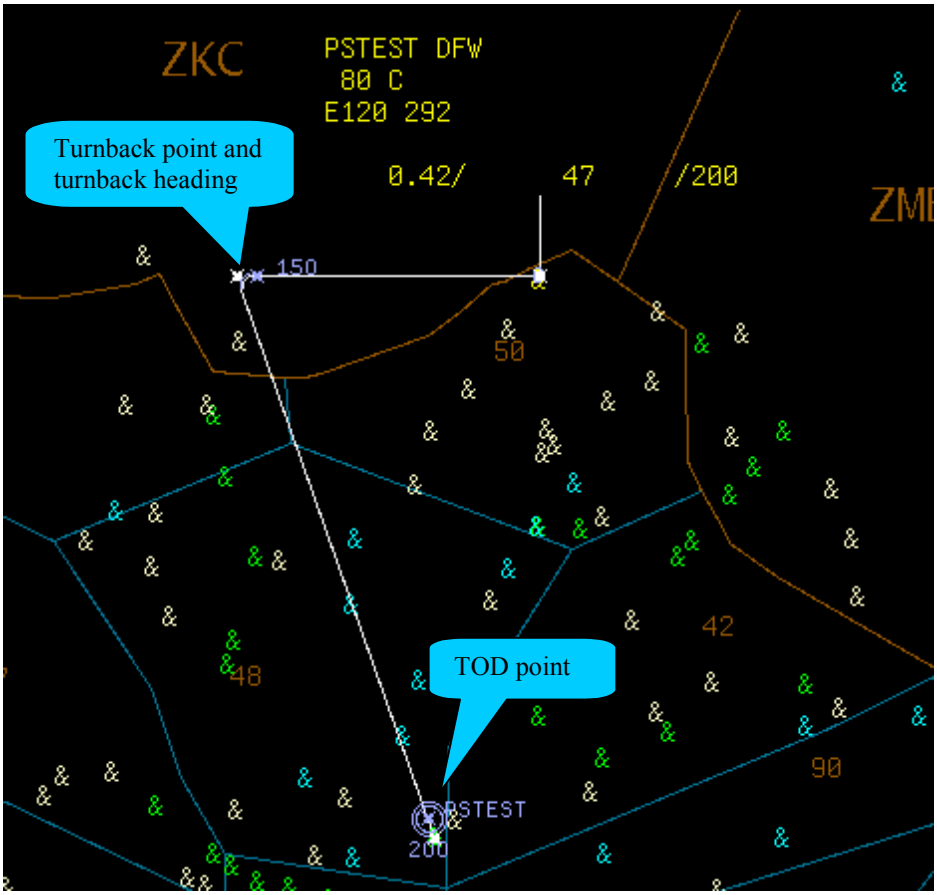


Figure 2-9 Path Stretch display

During the testing process a few problems were discovered with the code. When exercising the "Display PS Turnback as Absolute Time" mode from the <F1> panel, the PGUI crashed. Also, the aircraft followed the new heading for a while but then the

PFS_C process crashed. The first problem was debugged and a fix was implemented within the baseline. Originally a temporary fix was made to Seagull's local copy of CTAS to alleviate the second problem. The problem also occurred in the meet-time code and was fixed in that code. When the meet-time functionality was implemented, the Path Stretch problem was also fixed.

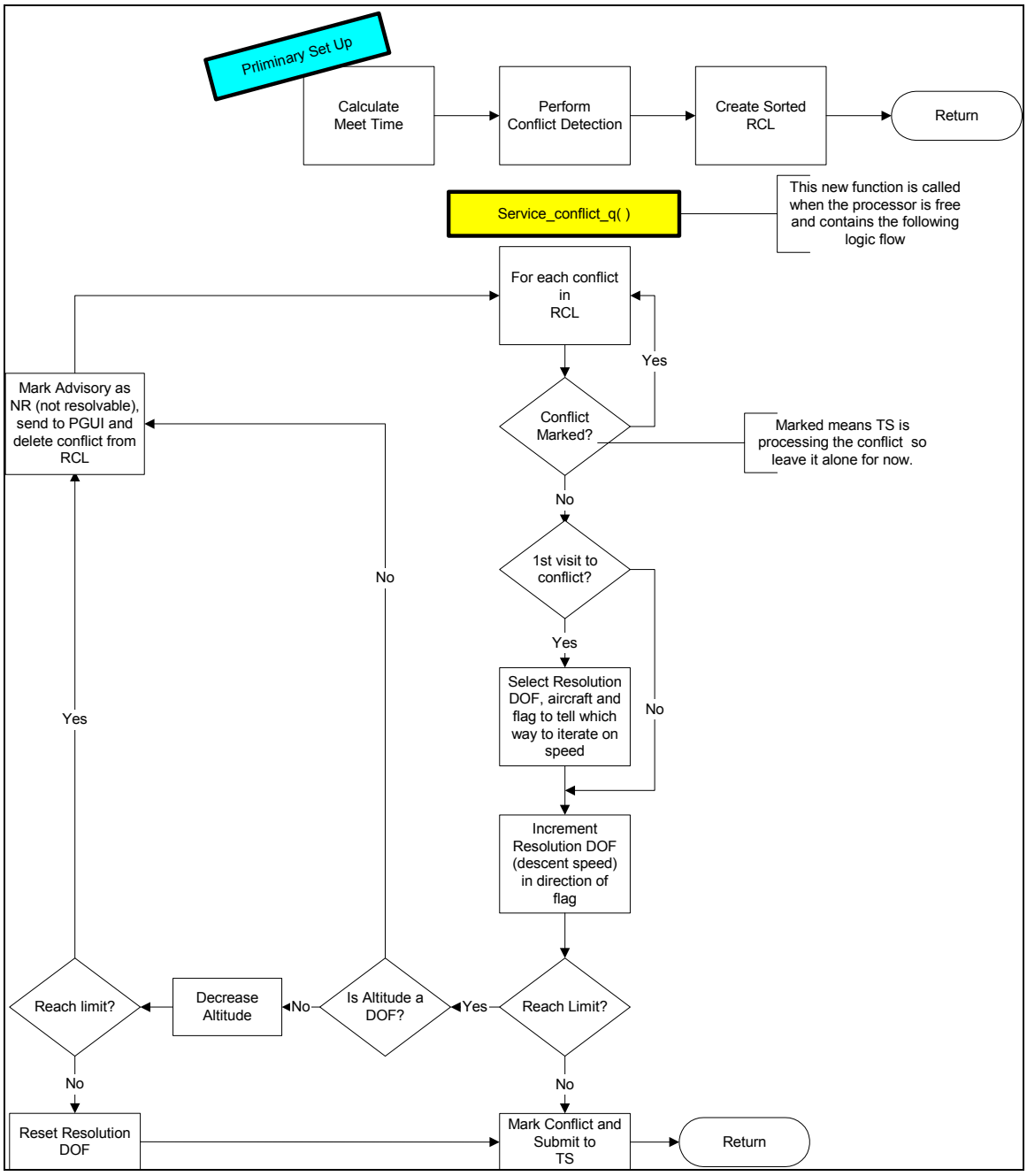
NASA personnel continue to try to get Path Stretch working with the new TS process. It was discovered that further changes to the new TS process needed to be made. Path Stretch now works when the PFS_C can come up with trajectories. However, there still are inconsistencies that are currently being evaluated.

2.5 Conflict Resolution

This task was defined as implementing the Slattery algorithm within the CTAS baseline. This algorithm was detailed in the research described in [SG94]. The algorithm was implemented in the original CAST version but was never carried over to CTAS. Seagull analyzed the Slattery-Green paper [SG94] and developed a design for implementing a version of the algorithm within the current baseline as documented in [DWS02]. The CAST architecture (which will not be outlined in this paper) is quite different from the current CTAS architecture. The current CTAS employs what is referred to as RAPS architecture.

RAPS architecture isolates the libraries that are common to both the RA and the PFS_C processes and their interaction with the TS process. The TS calculates trajectories and, in PFS_C, input to the TS is derived from analysis categories as well as other data. These categories are determined by the state of the aircraft and other global items when the radar update occurs. For example, questions such as: Is the aircraft in cruise or descent? Is the aircraft on its flight plan? Is the aircraft in an En Route center or TRACON? are asked and the proper category is determined by the answers. These categories then set degrees of freedom (DOFs) and their limits to be used during the trajectory processing. Currently there are no existing categories to handle the Slattery conflict resolution. Therefore, for EDA Build 2, we tried to resolve conflicts between arrivals within the same stream class by iterating on the DOFs using a non-RAPS approach.

The algorithm within the Build 2 architecture is displayed in the flow charts in Figure 2-10. The two main parts of this implementation are the service_conflict_q() function and the response to the TS regarding Conflict Resolution. The service_conflict_q() is called when the processor is free. It processes the conflicts that the Slattery algorithm is interested in one at a time. The response to the TS concerning Conflict Resolution iterates on the DOFs to solve the resolution while maintaining conformance with the scheduled time of arrival (STA).


Prliminary Set Up
Calculate Meet Time
Perform Conflict Detection
Create Sorted RCL
Return
Service_conflict_q( )
This new function is called when the processor is free and contains the following logic flow
For each conflict in RCL
Yes
Conflict Marked?
Marked means TS is processing the conflict so leave it alone for now.
No
1st visit to conflict?
Yes
No
Select Resolution DOF, aircraft and flag to tell which way to iterate on speed
No
Increment Resolution DOF (descent speed) in direction of flag
Reach Limit?
Yes
Is Altitude a DOF?
No
Decrease Altitude
Reach limit?
Yes
No
No
Mark Advisory as NR (not resolvable), send to PGUI and delete conflict from RCL
Reset Resolution DOF
Mark Conflict and Submit to TS
Return

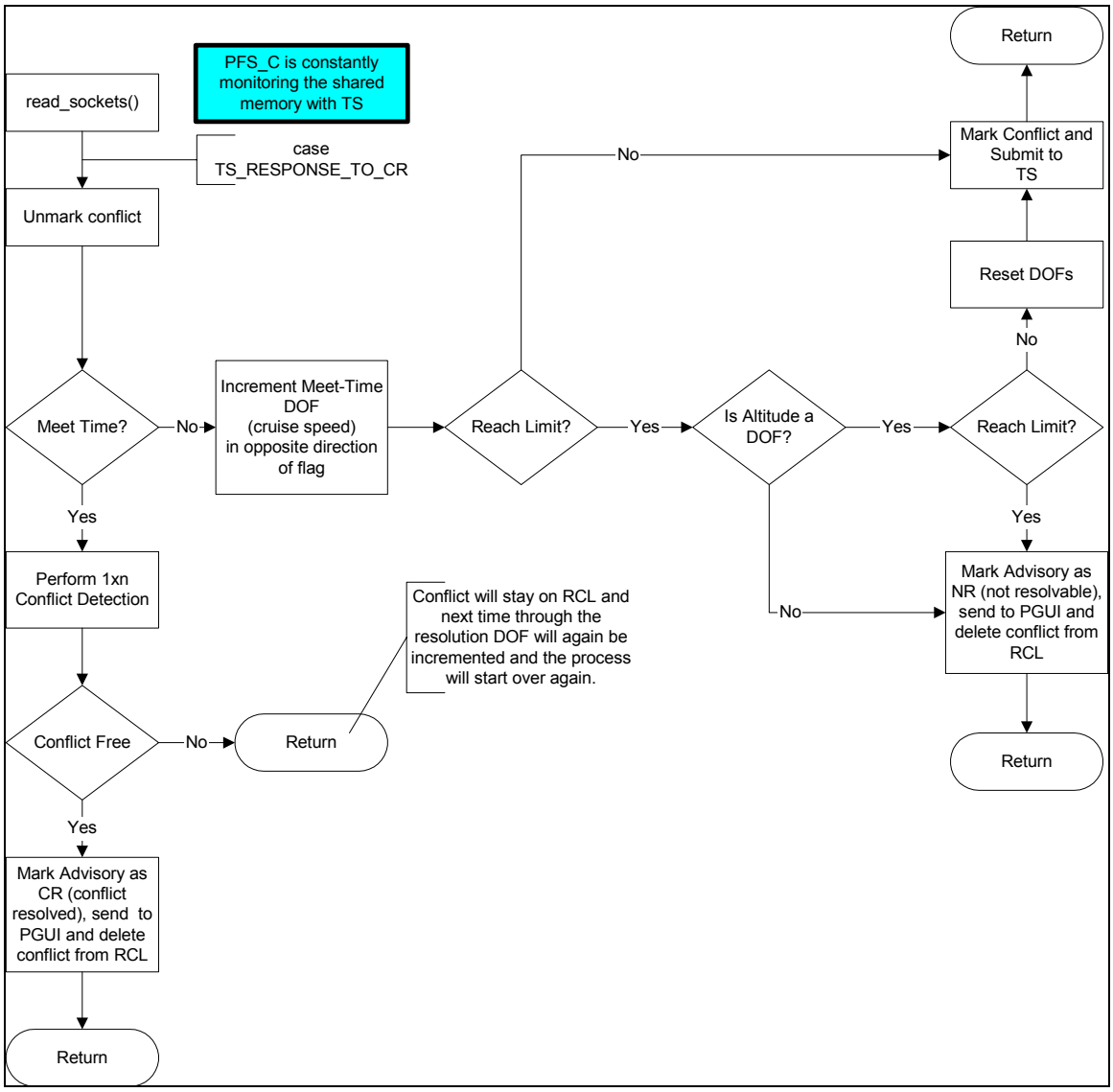


Figure 2-10 Slattery Algorithm within the Build 2 Architecture

To initiate conflict resolution, the <F1> panel of the PGUI was changed so that the user could turn conflict resolution on or off and select whether to allow altitude to be used as a resolution DOF. The user clicks on the Configure Conflict Resolution button that is shown in Figure 2-11 which brings up the panel in its default configuration shown in Figure 2-12. The user may now turn off conflict resolution altogether or just deselect altitude as a resolution DOF. Cruise speed and descent speed DOFs are both necessary for the Slattery algorithm, therefore we do not allow the user to turn these off.

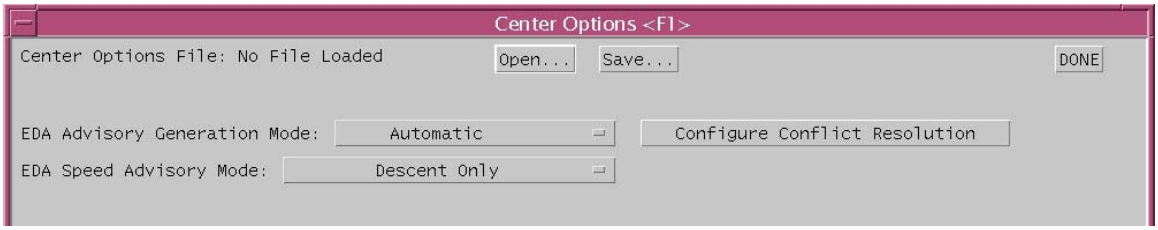


Figure 2-11 Conflict Resolution Access from <F1> PGUI Panel

Figure 2-12 Panel used to Turn Conflict Resolution On or Off and to Select Altitude as an Available DOF

At the current time, code has been written to implement the creation of the sorted RCL, selection of the DOF, selection of the aircraft to adjust and the determination of the flag that controls the way the DOF is adjusted. Shortly the service_conflict_q() structure will be developed.

3 Testing

The purpose of this section is to examine the effect of the EDA Build 2 implementation on CTAS and CTAS's current structure on the EDA Build 2 implementation.

3.1 Creation of Scenarios

A number of scenarios were created from two sources to perform the tests and evaluation.

3.1.1 Radar Data

Radar data were captured from a morning rush period on January 14, 2002, as shown in Figure 3-1. This file was used to create scenarios with one aircraft, a moderate sized group of ten aircraft all of the same stream class and a full-up scenario with arrivals at all of the metering fixes, departures and overflights. The data in these scenarios may be observed but cannot be controlled. They represent what actually happened during the time the data were captured and the flight paths of the aircraft are a result of the controller actions.

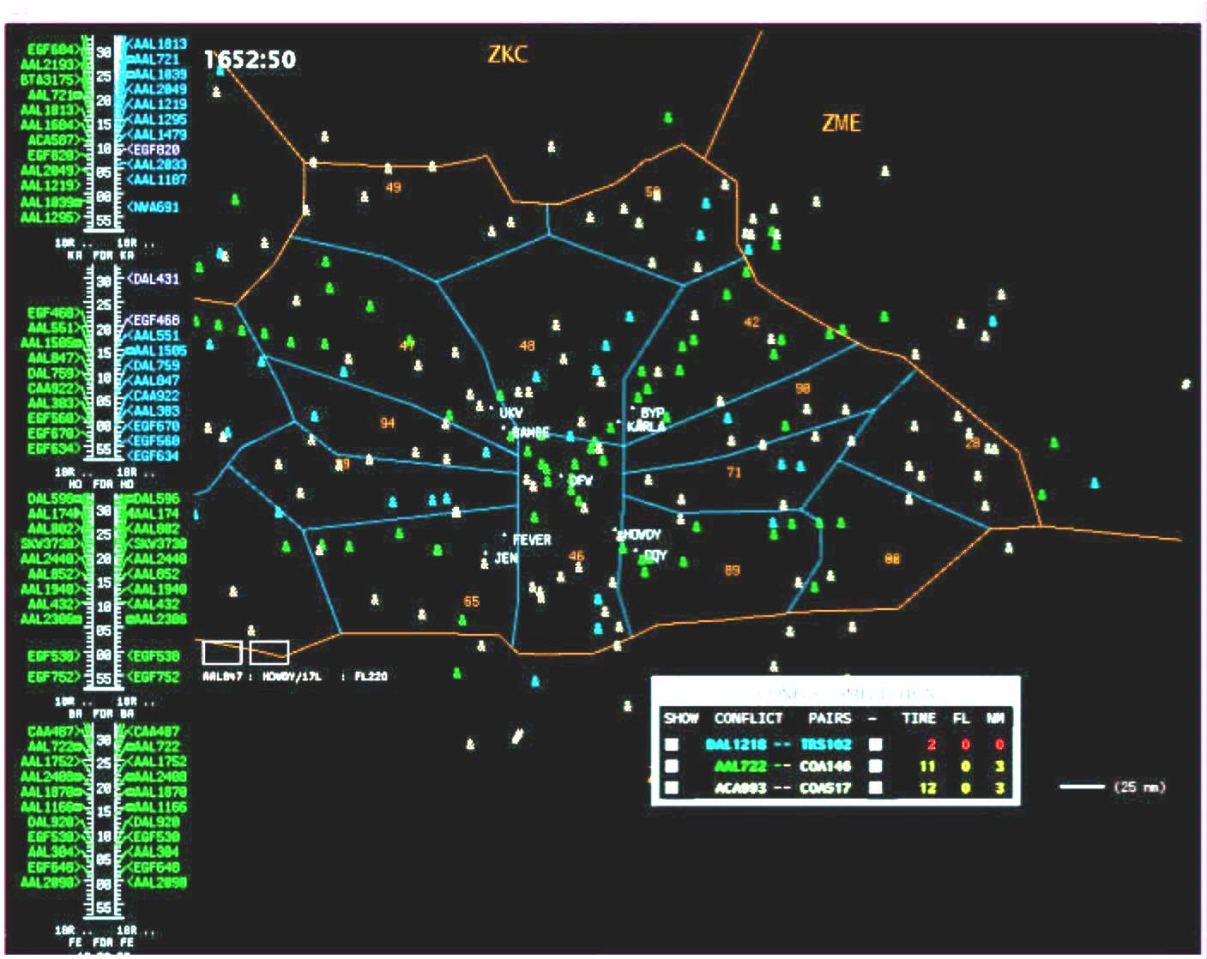


Figure 3-1 PGUI showing radar data from 14-January-2002

3.1.2 PAS Data

The Pseudo Aircraft Systems (PAS) tool was used to develop similar scenarios (1 aircraft as shown in Figure 3-2 and a group of ten aircraft all of the same stream class as shown in Figure 3-3). The aircraft in these scenarios may be controlled by the tool and therefore can be used to test the meet-time compliance.

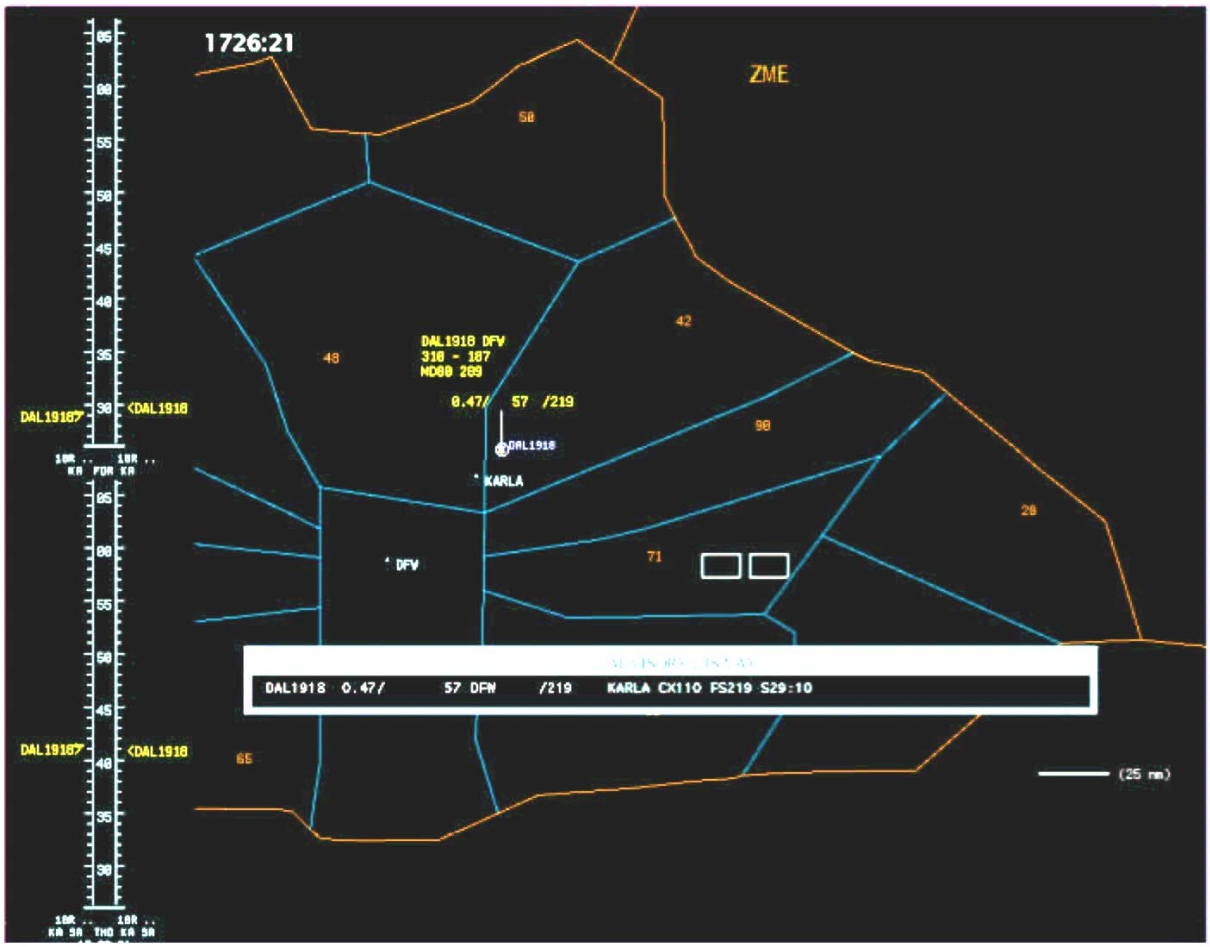


Figure 3-2 PGUI showing a single aircraft being controlled with PAS

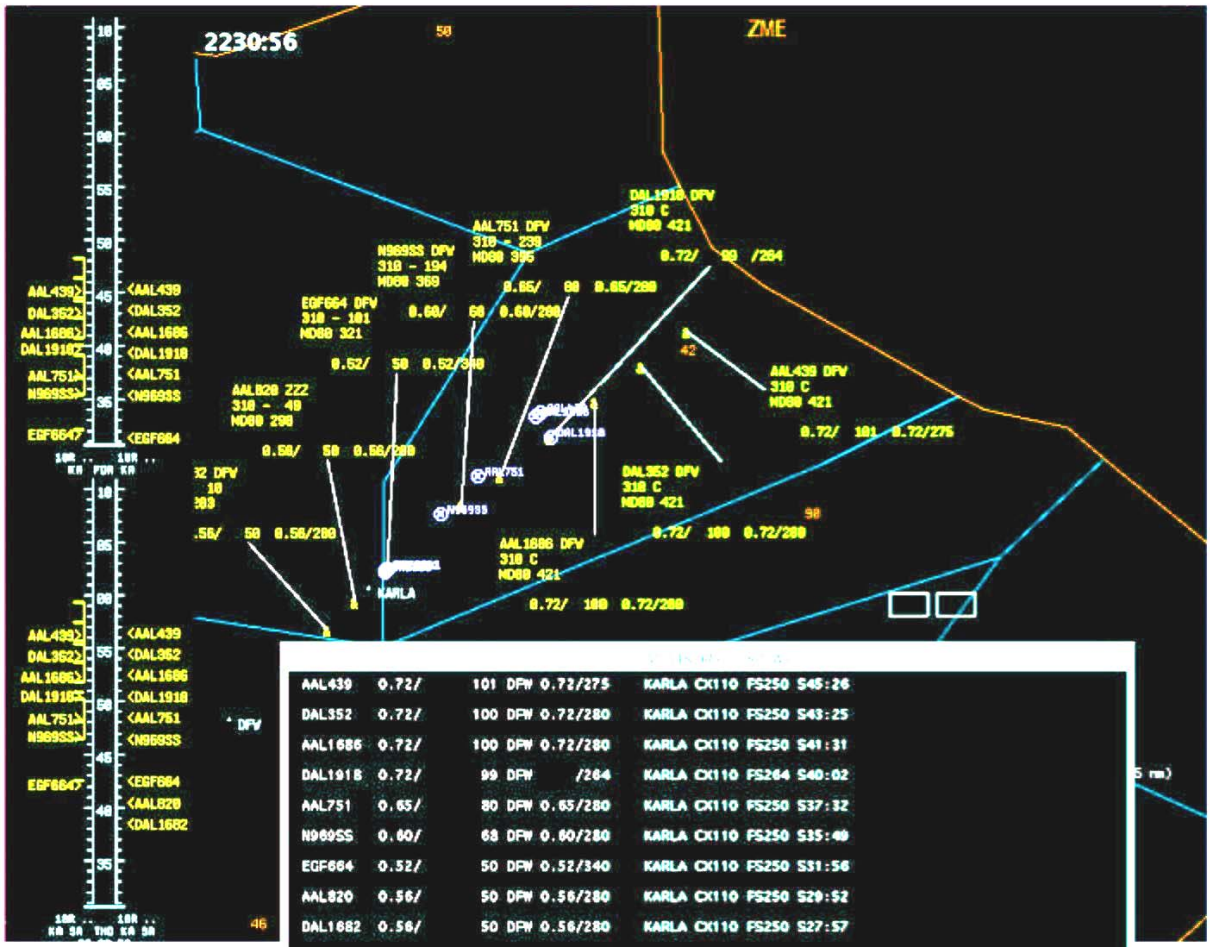


Figure 3-3 PGUI showing ten aircraft being controlled with PAS

3.2 Hardware

The validation and performance testing was conducted on Sun Ultra 5 computers running at 333MHz with either 512 MB or 256 MB of memory (machine names: heron, egret, pegasus, isle-royale, paoc) and two Sun Ultra 10s with 440MHz and 256 MB of memory (machine names: dove, flygirl). The CTAS processes were distributed on several computers, as shown in Appendix-A, for the validation and performance testing. CM was executed on paoc (360 MHz) for all trials. DP and PGUI were executed on egret (333 MHz) for all trials. ISM was executed on paoc (360 MHz) for all trials that used PAS. SIM_MAN and PILOT_MAN (PAS processes) were executed on heron (333 MHz) for all trials that used PAS. PFS_C and RA were run on different computers (with different CPU speeds) as shown in Appendix-A.

3.3 Evaluation of Accuracy

Our plan to test the accuracy of the meet-time advisories consisted of using the CTAS generated cruise and descent advisories as commands to aircraft via PAS and determine if the aircraft met the calculated STA. This was easier said than done.

Sections 3.3.2, 3.3.3 and 3.3.3 describe what was actually done for the record. After this work was completed, we were able to discuss our approach and results with Titan EDA team members, Bob Vivona and Husni Idris at Seagull Technology, Inc. on March 21st, 2002. They verified what we suspected. Our method tested the ability of PAS aircraft to fly a trajectory – actually comparing the trajectory generation capabilities of PAS and CTAS. A description of the method proposed by Titan that will actually test meet-time advisories is in Section 3.3.4.

3.3.1 Approach

The "Crossing Restrictions" (CRS) and "Top of Descent" (TOD) commands were the two PAS commands that were considered. The CRS command allows speed and altitude at the Metering Fix (MF) crossing to be specified. The TOD command allows a descent mach and speed to be specified. Steve Bayne (NASA Ames) suggested using the CRS command since the point in space (e.g. metering fix (MF)) can be specified along with the desired crossing altitude and speed, whereas the TOD command does not shoot for a point in space. However, since the point of this exercise is to evaluate the accuracy of the meet-time algorithm, we decided to use the TOD command which does attempt to fly a standard trajectory (initially hold mach and then holding speed). According to Steve Bayne, the CRS command uses a 2:1 or 3:1 descent ratio (nautical miles to 1000 ft) – basically allowing the AC to fly whatever trajectory is necessary to meet the crossing restrictions (time, speed and altitude). As an alternative to PAS, controller inputs directly to PGUI were briefly considered as an alternate to PAS but rejected as that input uses radar tracks so the actual trajectory is not affected. Although PAS has limited ability to correctly fly a specified trajectory, PAS was the only tool available for this study.

3.3.2 Validation Testing Procedure

The procedure for testing both descent-only and cruise-only mode was to input an "aircraft list" file and "auto command" file into the Simulation Manager (SIM_MAN process). These files contained the list of aircraft (with associated information) and any PAS commands that could be known prior to the run (e.g., the STAR command), respectively. During descent-only mode runs, the TOD command (including advised descent mach/CAS) was entered into PAS prior to the TOD point and this command was automatically executed when the aircraft reached the TOD point. The advised descent mach/CAS could not be known prior to the run, as it is dependent on the STA. During the cruise-only mode testing, a cruise mach/speed command was issued well before the TOD point (based on the displayed cruise advisory) and then the TOD command was entered and executed as in descent-only mode.

Dynamic Planner (DP) was included for all validation (and performance) testing, because this caused more consistent testing. Disabling the connection between DP means that TMA will not be generating STAs but they will be set equal to ETAs.

Test runs for single aircraft (of MD80 aircraft type) and a stream-class (of all MD80 aircraft type) were conducted for the meet-time validation testing.

Although some runs were conducted without changing STA, STA was generally changed (via the time line in PGUI) for the reported results since changing the STA would cause

the meet-time algorithm to be invoked. For the cases where STA was changed it was adjusted by moving the STA as displayed on the PGUI to a value that was within the ETA bracket.

The input parameters for the validation test matrix were; mode (cruise-only or descent-only), initial distance from MF, STA, advised mach/speed.

DFW was the destination airport for all testing. The single aircraft and the stream-class tests used KARLA (the northeast MF) as the MF en-route to DFW. The operationally agreed-to crossing altitude for KARLA is 11,000 ft.

3.3.3 Validation Testing Results

Although use of the CRS command was discontinued (the results are not described herein) the CRS command was generally observed to yield a MF crossing within the time tolerance (+/- 20 seconds) at the correct altitude.

The summary of the inputs and results for the single-aircraft validation test runs, which used the TOD command, are shown in Appendix-B. "Crossing time delta" is the difference between the changed STA and MF crossing time, unless STA was not changed and then pre-TOD STA is used in the difference calculation.

The effect of distance from the MF was not investigated for descent mode, as no changes are made during the cruise portion of the flight – all inputs to the trajectory are made at the TOD point and affect the descent. There appears to be a correlation between how much the STA was changed and the crossing time, where the further from the initial STA the STA is changed (delayed in time) the earlier the AC crosses the MF with respect to the specified STA. The trial-6 on 27 Feb (where a different hardware configuration was used) does not fit as well into this trend. This may be due to the faster CPU for the pfs_c process in that run. In Descent mode, if the initial STA was changed less than -30 seconds, the TOD command yielded a MF crossing within the time tolerance (+/- 20 seconds) but the crossing altitude was off (by 3-4 thousand feet).

In Cruise mode, the MF crossing time was generally within the +/- 20-second tolerance, but the crossing altitude was too low. Although investigated, no effect due to initial distance from the MF was found on MF crossing. The crossing altitude was always below the correct crossing altitude of 11,000 feet.

The summary of the inputs and results for the stream-class validation test runs are shown in Appendix-C. "X" indicates that that there was an input error and the results are incorrect. "?" indicates the value was not recorded. "??" indicates that the value was not displayed. For the two cruise-only mode cases, the pre-TOD advised cruise mach/CAS used a cruise speed input well before the TOD point. The stream-class results serve to show that nothing unusual occurred as compared to the single aircraft cases. STA was changed for a few of the aircraft, which also did not cause any unexpected behavior.

3.3.4 Proposed Meet-Time Advisory Testing

Approach:

First, determine the correct descent/cruise CAS to meet a user specified STA and then determine if the meet-time algorithm produces the correct (CAS) advisory. Using the standalone TS, vary CAS from min to max and find the ETA for each CAS. Then, using CTAS, enter new STAs over the same region and see if the same speed advisory is obtained. This will verify the accurate repeatability of the meet-time algorithm and reveal if there is a 'better' answer – the algoithm picks the first CAS it finds (within the +/- 20 second tolerance range), but there may be a second advised CAS in this range that is actually better.

Methodology:

Use PGUI to produce a trajectory dump file. Input this file (information) into the stand-alone TS to produce a mapping between CAS and time. Run CTAS (with scheduler off – PGUI F2 menu: uncheck 'TMS Scheduled Times'), dwell on AC (in PGUI) and press shift-S to enter the STA (format: hr:min:sec). Verify that for the STA entered, the correct CAS (from the CAS to time mapping) appears in the advisory.

3.4 Evaluation of Performance

Test runs for a single aircraft, and full-up (many aircraft - actual data playback file, with radar tracks) were conducted for the performance testing.

Dynamic Planner (DP) was left on for all performance testing.

The configurations of process distribution on the available hardware are described in Section 3.2 and shown in Appendix-A.

3.4.1 Performance Testing Results

The results of the performance runs are shown in Appendix-D.

The speed of calculating the meet-time advisories appears to have no significant effect on EDA Build 2. However, it is still recommended that the PFS_C process be run on the fastest computer available.

Comparing the cruise-only mode trials (28 February, 1 March) that used the radar input file, the average duration of the meet-time algorithm does not seem to have a significant correlation with processor speed or the number of RA processes being executed (cross reference to Appendix-A), or whether cruise or descent mode was used. However, there does appear to be an effect of processor speed on the average meet-time calculation in descent-only mode, where the average meet-time calculation times for trial-1 on 28-Feb and 1-Mar (which were conducted with PFS_C on a 440 MHz CPU) are noticeably less than the average meet-time calculation times for trial-3 on 28-Feb and 1-Mar (which were conducted with PFS_C on a 333 MHz processor). The network activity was of interest, but there is no noticeable difference between the four trials which were run in the mid-afternoon – usually a busy period (28 February) and the four trials which were run early in the morning – usually a quite period (1 March). These trails did not put a significant load on the network.

3.5 Evaluation of Path Stretch Initiation Automation

During the current effort, the possibilities for providing automated Path Stretch initiation were considered. Our findings are documented in this section.

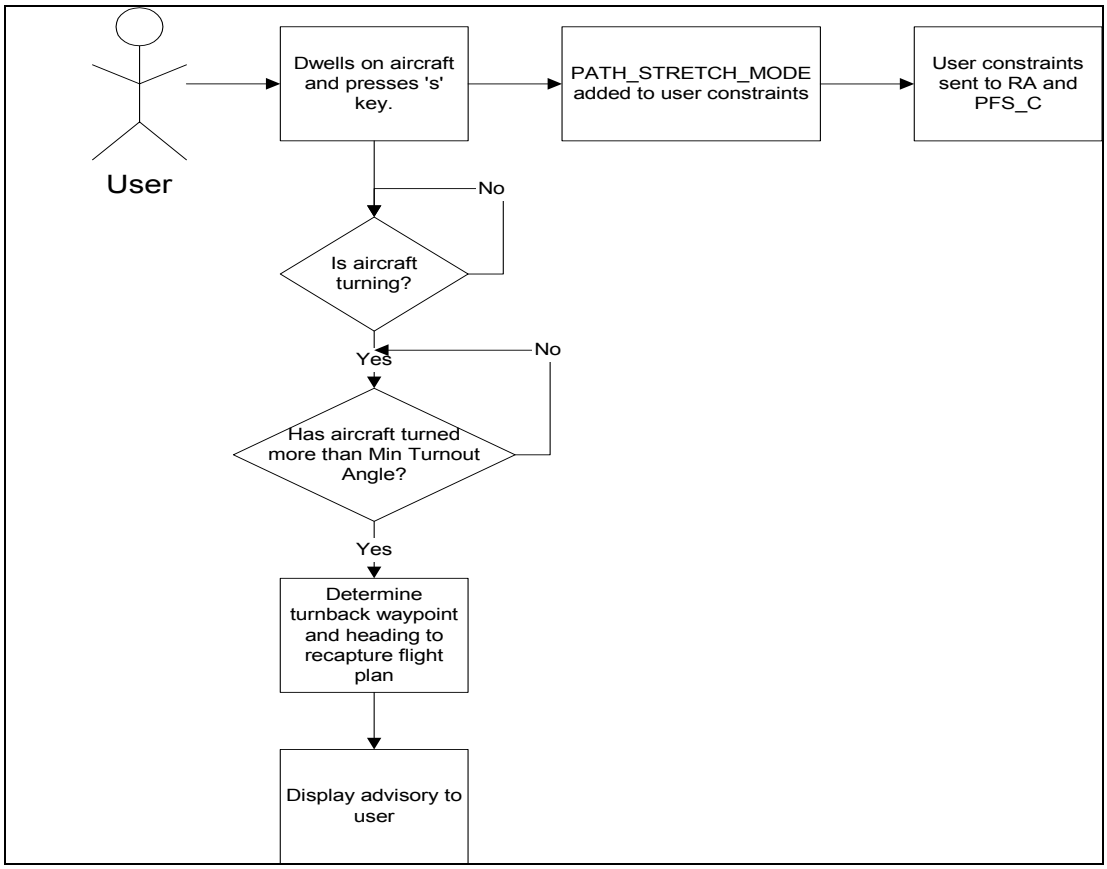


Figure 3-4 Path Stretch initiation in current CTAS baseline

As shown in Figure 3-4 in the current CTAS baseline, the initiation of Path Stretch is a manual process. The controller first has to put the aircraft into Path Stretch mode by dwelling on the aircraft icon and pressing the 's' key. This action adds PATH_STRETCH_MODE to the user constraints and then sends the user constraints to the RA and PFS_C processes. As an indicator to the user, a message saying the aircraft is in Path Stretch mode ("Path Stretch on") appears under the scratch pad on the PGUI. Also, a 'PS' appears on the sequence list if the user had selected that advisory to be visible and 'PS' appears in the advisory on the data tag.

Next the controller has to turn-out the aircraft (issue a heading command whose angular difference from the current heading to a capture waypoint is greater than the minimum set in the <F1> panel shown in Figure 3-5). If no capture waypoint is set, the flight plan meter fix is used to determine the turnout condition. In the testing mode, this process is accomplished by creating and vectoring a fake aircraft or by having a pseudo-pilot running PAS turn the aircraft.

Figure 3-5 Minimum Turnout Angle Setting on <F1> Panel on PGUI

RA and PFS_C know about the minimum turnout angle due to a PGUI_MIN_TURNOUT_ANGLE message sent by PGUI either at startup or when the value is changed.

The software then determines if the aircraft is eligible for Path Stretch by determining if it has reached its required minimum turnout angle. The difference between the aircraft's current heading and the heading from the aircraft's current position to the capture waypoint is compared to the minimum turnout angle during this test. If it is greater than or equal to the minimum turnout angle, the software declares that the turnout is complete. The software then determines the time and distance to the turnback point and a heading at the turnback point to recapture the flight plan so that the STA may be met.

To automate the initiation of this process, one can consider at least two possible scenarios; one semi-automatic and one automatic. In the first semi-automatic scenario, the user would still press the 's' key to place the aircraft into the Path Stretch mode and then the algorithm would start calculating the turnback point. With the second option, the user would not have to press the 's' key. The software would determine that the STA cannot be met with speed and altitude changes alone and would automatically place the aircraft into Path Stretch mode. In both cases, the algorithm would follow the same logic from this point.

Currently in CTAS, the way to get an aircraft to turn is to create an auxiliary (AUX) waypoint and update the aircraft's flight plan. Therefore, to get the aircraft to turn, it is suggested that a temporary AUX waypoint be created at some small distance from the aircraft's current position at a heading that is the current heading plus or minus the minimum turnout angle. This distance could be set by a user input found on the <F1> panel. The direction of the turn (e.g. adding or subtracting the minimum turnout angle) could be set by a rule-of-thumb or both could be set and later compared to see which meets the criteria best. A discussion of "best" can be found later in this section.

Let's look at Figure 3-6 to better understand the proposed algorithm. The current algorithm assumes it knows the new heading of the aircraft. The new algorithm would create an AUX waypoint 10 (or whatever distance selected by the user) miles out at the minimum turnout angle and turn both right and left and calculate new turnback points. It would then go through a table of constraints to see if it should keep the result or discard it. Some of the constraints could be:

- not crossing a sector boundary
- not creating a conflict with another aircraft
- keeping the turnback angle less than some maximum
- not being able to capture the next waypoint

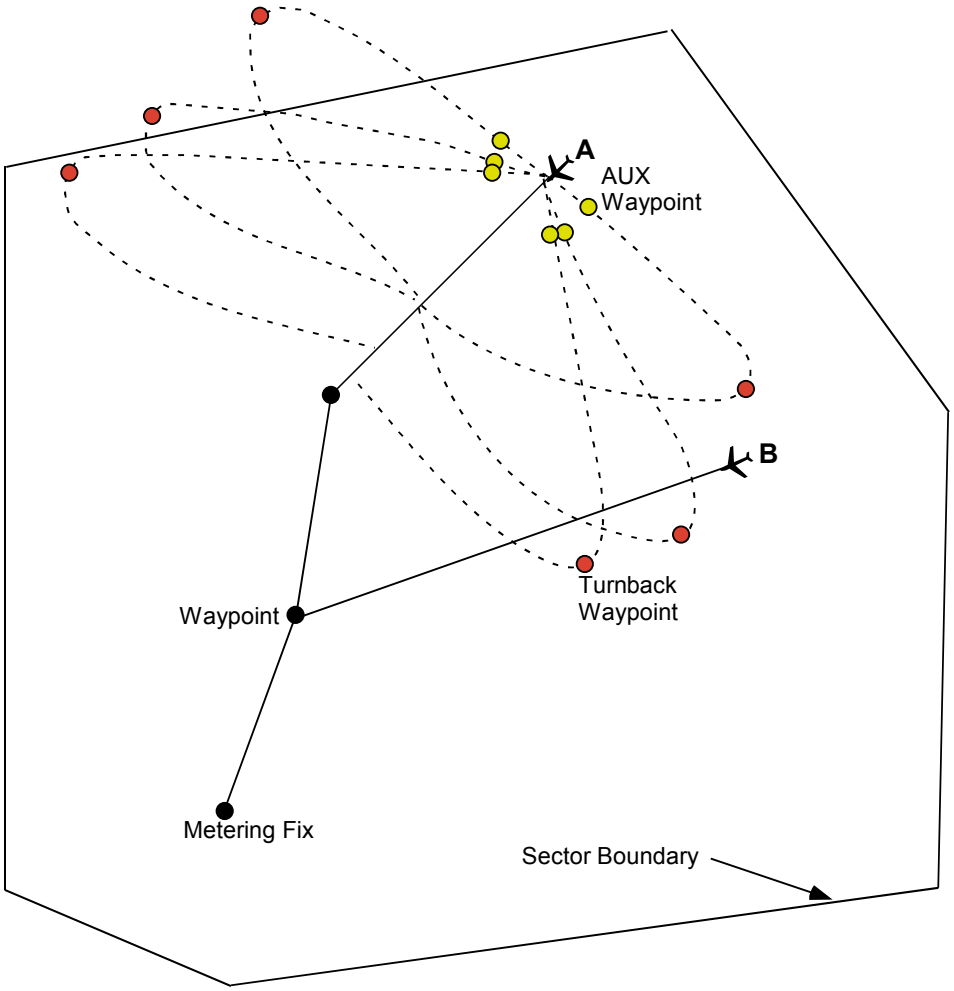


Figure 3-6 Possible AUX Waypoints for Auto Path Stretch

If both options were discarded, the algorithm would increase the heading by a given delta of 10 degrees, create the new AUX waypoints, calculate the new turnback waypoint and perform the test again. This could continue until some maximum turnout angle is reached, say 90 degrees. (After that a holding pattern may be a better solution.) If both the right and the left options met all of the constraints then an established rule-of-thumb (e.g. always choose the right solution) would be used to determine which solution to pick.

Once the solution was determined, the turnback waypoint would be added to the flight plan and the advisory would be presented to the controller as is shown graphically in Figure 2-9 and in the data tag as shown in Figure 2-5.

3.6 GUI Feedback

So far, after demonstrations to various NASA personnel, there have only been minor requests for changes to the EDA Build 2 PGUI graphic enhancements. Some may be fixed by changes to the EDA default files; others will need further investigation into CTAS GUI functionality.

Sometimes there is a delay when moving the STA on the timeline and the STA appears to be moved to a position relative to the tail of the cursor and not the head. The EDA Build 2 software team used the same procedure for this move action as is used to move data tags. The team will investigate methods to improve the STA movement response and continue to investigate other causes to the delay.

3.7 Evaluation of Build 2 Software Architecture

The current CTAS architecture, specifically the PFS_C architecture, was evaluated to see how well it fits the Build 2 design and how well it will fit the Build 3 design as outlined in the *En Route Descent Advisor (EDA) Build 3 System Specification* document [SSS]. Specifically, we looked at a) the predictability of receiving data updates in a time frame needed for EDA and b) the timing of the calculations of conflicts so that resolutions can occur.

3.7.1 Arrival of new data

The CTAS Input Source Manager (ISM) receives most aircraft updates at a 12-second update rate. There may be some maximum read capacity but all data for a one radar sweep are received within 1 to 2 seconds. The Communications Manager (CM) process handles the data almost instantaneously and sends the track data to the Route Analyzer (RA) to determine ETAs. However, the speed at which the RA calculates the ETAs is variable and can depend on many things, such as the number of aircraft to be analyzed and the number of RAs doing the processing. Once the ETAs are derived, they are passed to the CM and this process distributes the information to other processes.

For EDA, the Profile Selector – Center (PFS_C) process determines conflict-free aircraft trajectories that meet any meet time constraints. Within this process the trajectories are probed for conflicts and probed to see if they are eligible for Direct-To advisories. For EDA Build 2, advisories are generated for arrival trajectories so that the trajectory may meet a given time at the Metering Fix. And, finally, conflicts involving metered arrival aircraft are resolved within this process.

The PFS_C process receives analysis packets from the RA process for active flights. The number of runways analyzed for that flight determines the number of packets for each flight because an ETA for each runway is generated per aircraft. The data are received in a RA_ANALYSIS_PACKET message.

We determined that it is important to understand when the trajectory data are being updated with respect to when the trajectory advisories are determined, when the trajectories are being probed for conflicts and when resolution advisories are determined.

The PFS_C process is a single process with no threads. In some ways it works like a multi-threaded process in that there is a sharing of processing time. Some activities release the processor when they do not need it any more and other actions can continue or start. The reading of the RA_ANALYSIS_PACKETS and processing of the trajectories take place with this time share method. Their actions take place when the processor is available. Other activities take place at a scheduled time in the cycle. Conflict probe is an example of this kind of activity. Currently (developed for the Direct-To/Trial Planner tool), the cycle to process these scheduled activities is broken up into 12 frames to

emulate the 12-second radar update cycle (1 frame per second). However, the frames are not synched to the receiving of the data. The trajectories are probed for conflicts in Frame 6 and the probing is forced to end if not finished by Frame 9. There is a second probing of current data in Frame 0 to emulate a 6-second update.

In order to understand when data are updated versus when other activities take place we analyzed a number of items.

- How related are the frames to the supposed 12-second radar update?
- How many analysis packets are actually received in a 12-second update and how many analysis packets are received within each Frame?

We ran a single arrival flight to gather initial information and then ran a scenario with 20 arrival flights. Upon study we found that for the 1-flight case the RA_ANALYSIS_PACKET is not received exactly every 12 seconds, but every 11 to 14 seconds. Therefore, the data are not received in the same frame every update. It was assumed that the conflict probe would be working on new updated data; however, from this analysis we find that that is not correct. The conflict probe may be working on data that, in the worse case, is 11 seconds old. Figure 3-7 shows the relationship between the arrival of the data to the current frame for the 1-aircraft case. An "ideal" scenario is also plotted for comparison in this figure, where the aircraft would always be updated before conflict probing takes place.

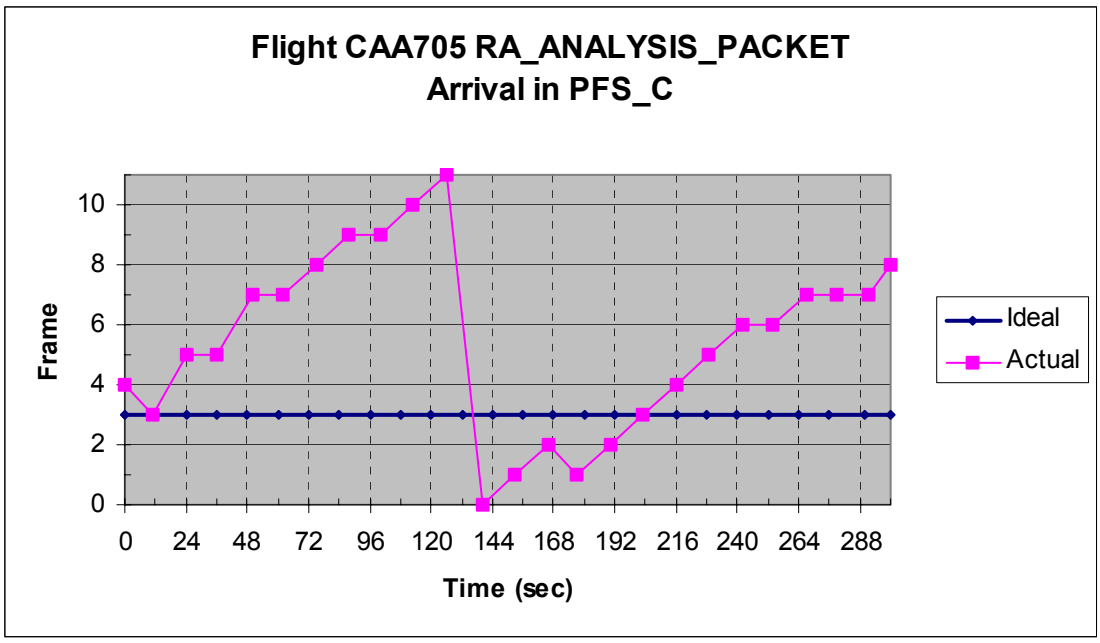


Figure 3-7 Relationship between the arrival of flight data to PFS_C and the current frame (1-aircraft case).

When analyzing the 20-aircraft case, we noted that a similar pattern develops after an initialization period. Most of the aircraft are updated on the 11 to 14 second cycle. The exceptions are dropouts and when an ADD_FLIGHT_PLAN message is received. In the latter case an RA_ANALYSIS_PACKET message immediately follows for the flight.

To determine for which aircraft RA_ANALYSIS_PACKETS were actually received during the twelve-frame cycle and in which frames the packets were received, we analyzed the 20-aircraft case over a few different cycles. These cycles represent the twelve frames that try to emulate one radar sweep. Figure 3-8 displays the results of this analysis. The frame in which the aircraft tracks are received is shown along with the frames in which the RA_ANALYSIS_PACKETS were received. This figure shows that we cannot count on receiving an RA_ANALYSIS_PACKET for each aircraft for which we have a track, shows that we do not instantaneously receive the RA_ANALYSIS_PACKET after getting the tracks and shows that there are variances from cycle to cycle. Other studies point to a better response using more RA processes and running on faster machines; however, unpredictability is still common.

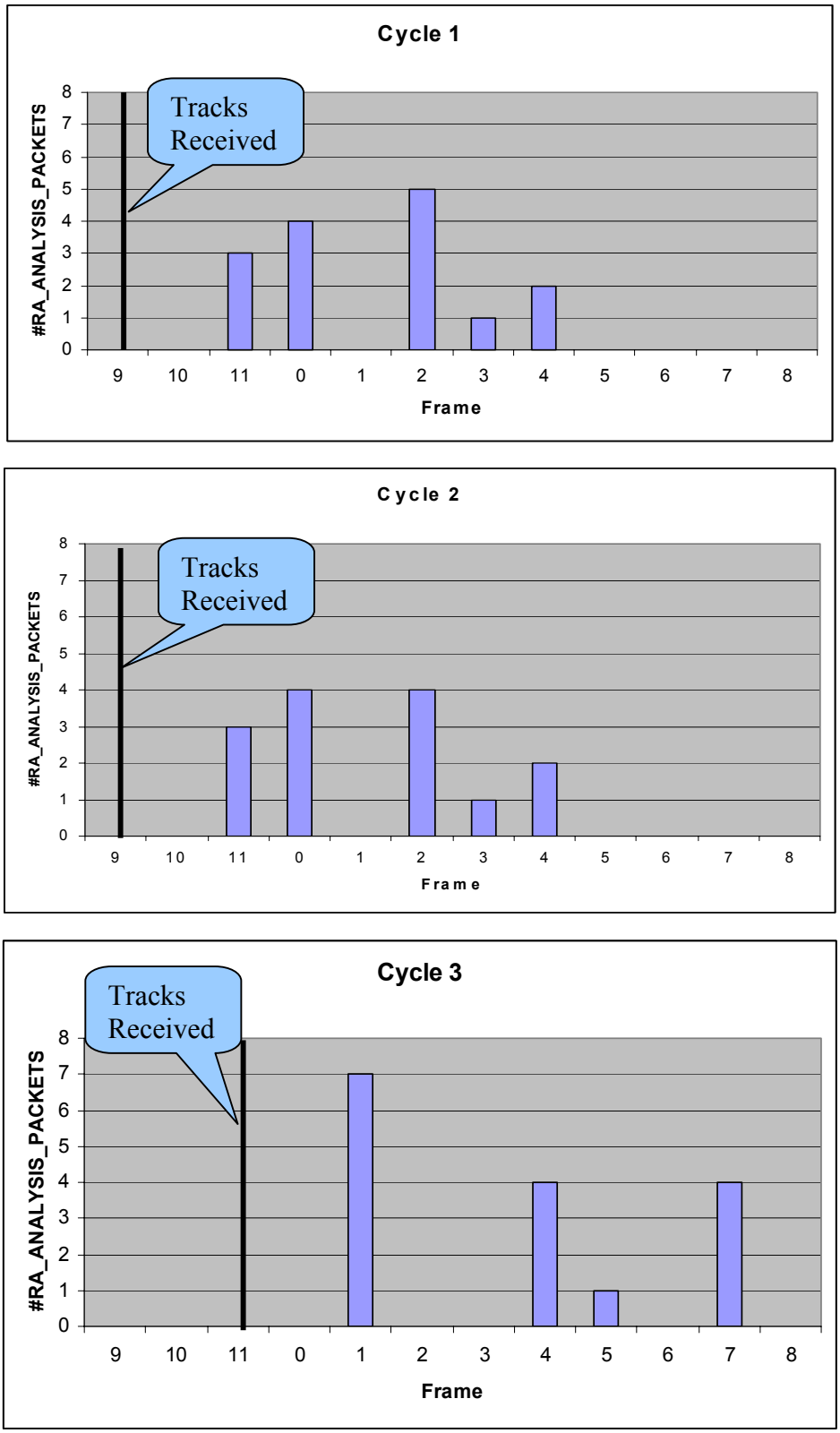


Figure 3-8 Number of RA_ANALYSIS_PACKETS received each frame during a 12-frame cycle.

Now what do these analyses mean to the current EDA Build 2 architecture? We cannot count on having all aircraft updated consistently before they are probed for conflicts and the conflicts are resolved. New analysis packets may be received between the conflict probe and the conflict resolution if the processor is released to read new messages. Therefore:

1) We must be sure that the data we are using to resolve the conflicts are the same data we used to determine the conflicts.
2) Since conflict probe is slated to be performed at most twice a sweep, we cannot provide resolutions at a higher frequency.
3) We must determine if the Frame method of handling conflicts is efficient for EDA needs.
4) We could consider synching the frames to the track message or make sure we have all of the aircraft before processing.

3.7.2 Architecture changes for improved Conflict Resolution implementation.

During testing, as shown in Section 3.7.1, we found no correlation between frames and when the analysis packets from the RA process were received. We also found no correlation between frames and when the tracks were received as shown in Figure 3-9. Although some runs did show a correlation (of frames with tracks), as shown in Figure 3-10, this correlation does not always have the tracks arriving in the same frame – it varies from run to run, and the correlation is not as desired. We would like all of the tracks to arrive in frame #1. Therefore, it has been proposed that the frame initiation should be synchronized more closely to when the system receives new track messages to try and make sure there is enough time to perform conflict detection and conflict resolution on updated aircraft data. It also would be helpful to monitor when tasks (e.g. conflict detection, conflict resolution) are finished and bump the frames so that processing time is not lost. The desired frame correlation for the various events is shown in Figure 3-11, for a hypothetical case.

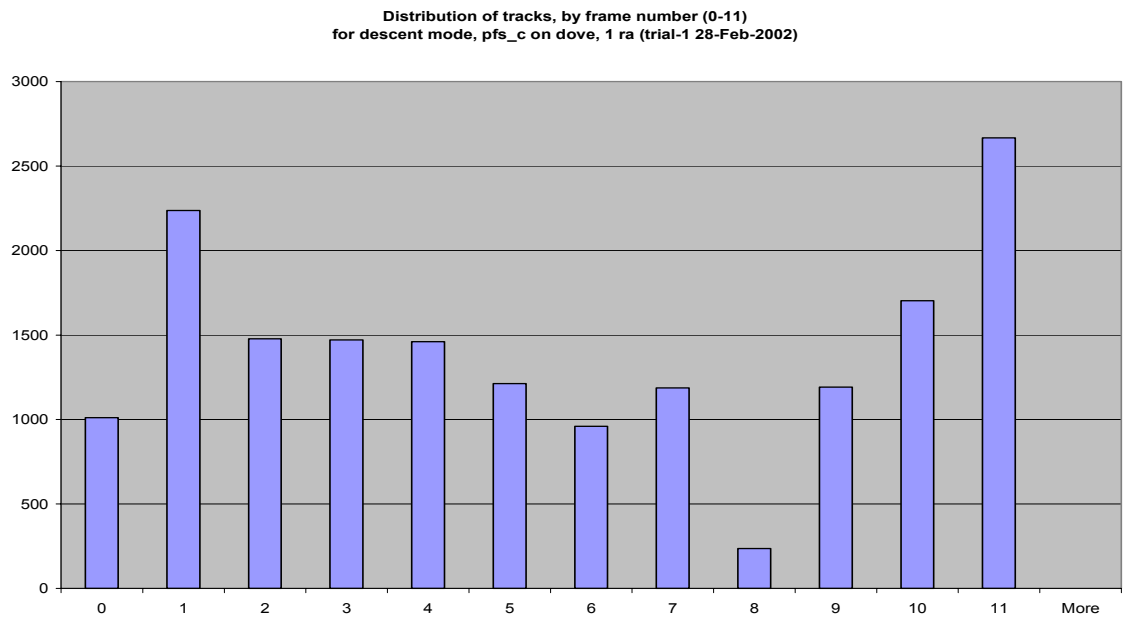


Figure 3-9 Distribution of radar tracks by frame number - uncorrelated

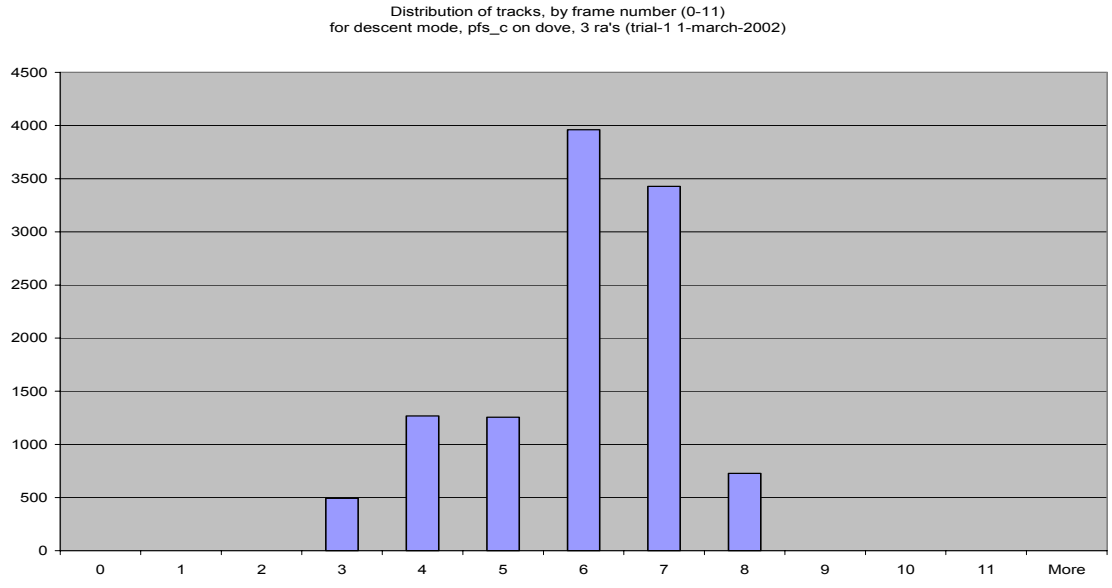


Figure 3-10 Distribution of radar tracks by frame number - correlated

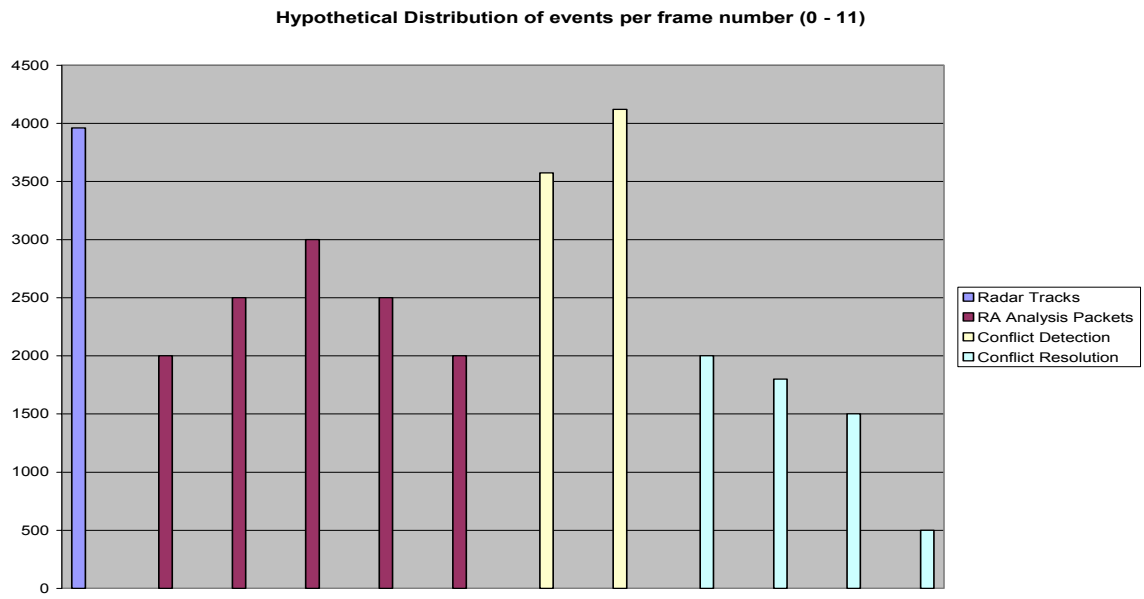


Figure 3-11 Hypothetical Distribution of events during the 12 frames

CM receives the track data all at once or at least within about 3 seconds. Then RA must process the tracks and that can take awhile, depending on how many RAs are running within the simulation and the speed of the processors on which the RAs are running. If we synchronize the frame update to when the track message is received, we still do not know that we will receive all of the analysis packets we need to process the trajectories for conflict detection and resolution before the frames that perform the work. This will have to be investigated with further testing once the new code is implemented. If PFS_C does not receive enough aircraft updates in time, other strategies will have to be considered. If we know the arrival aircraft that are in conflict, we could monitor the arrival of the analysis packets for these aircraft and postpone initiation of conflict detection and resolution until after they are updated.

Once the conflict resolution algorithm is implemented, detailed studies must take place to determine the solution that produces the most accurate results.

4 Summary and Conclusions

The task of implementing CAST functionality into the CTAS baseline has provided a good start to adding EDA functionality to CTAS. We now have some meet-time advisories being calculated and presented to the user in various ways, we have a timeline with adjustable STAs and we have the start of automatic conflict resolution for metered arrivals. We have also learned a lot about the current state of the CTAS baseline. The new information included:

- Meet-time was no longer being calculated using the new TS.
- Path Stretch did not work correctly with the new TS.
- New analysis categories needed to be added for each of the meet-time modes.
- The current RAPS architecture and specifically, the PFS_C architecture makes implementing the Slattery conflict resolution algorithm very difficult.
- There is no correlation between when PFS_C receives updated aircraft data and when conflict detection occurs.

Looking at the EDA Build 2 areas individually, we summarize what has been accomplished and what still needs to be done in the near future.

4.1 Path Stretch

4.1.1 What was accomplished

NASA personnel have worked with the CTAS software group to get Path Stretch working sometimes. Seagull has developed code to show the Path Stretch advisory in the data tag and has observed the ETA to come close to meeting the STA on the timeline once the advisory has been calculated.

4.1.2 What needs to be accomplished

- Make sure that when Path Stretch is being used as a meet-time advisory that the speed cases are not being used.
- Investigate why the turnback point keeps being recalculated after a certain time and fix the problem.
- Investigate why a new path calculation is not found when the STA is set to a new value after the initial setting and fix the problem.
- On the PGUI, turn off the time bracket when an aircraft is in the Path Stretch mode.

4.2 Meet Time

4.2.1 What was accomplished

Thanks to the work by the RTO63 group (Titan/SRC), two meet-time modes, Descent Only and Cruise Only, have been implemented. This work involved the creation of new analysis categories and changes to PFS_C to iterate on responses from the TS until the STA is met. Seagull has written code to display the advisory in the data tag and has initiated detailed testing of the meet-time results.

4.2.2 What needs to be accomplished

- Continue to test the implemented modes to make sure that the advisories that are calculated are the ones expected for the given STA.
- Make changes in the current implementation in response from CTAS RAPS software lead.
- Add new analysis categories for other meet-time modes.
- Implement other speed modes and then test them.

4.3 Conflict Resolution

4.3.1 What was accomplished

Seagull has implemented the GUI interface and message passing for the control of the initial Slattery algorithm. We have also coded the initialization of the Slattery algorithm within the PFS_C.

4.3.2 What needs to be accomplished

- Continue to try to implement the Slattery algorithm within the current PFS_C architecture.
- Test conflict resolution.
- Start extending Slattery to work with overflights

4.4 Architecture

4.4.1 What was accomplished

Seagull and RTO63 group have spent a great deal of time investigating the RAPS and PFS_C architecture with respect to both meet-time and conflict resolution and have designed changes to support EDA needs.

4.4.2 What needs to be accomplished

- In PFS_C, synch frame initiation to the receiving of track data.
- Try upping the frame count once the action that was assigned to that frame has finished.
- Try removing frames and work from a procedural event driven architecture.

4.5 GUI

4.5.1 What was accomplished

Seagull restored the ability to move the STA on the timeline, added the ability to select an aircraft from the timeline and coordinate the dwelling on an aircraft to the highlighting on the PGUI and on the timeline. As stated in 4.2.1, we added the advisory to the data tag and as stated in 4.3.1, we added the conflict resolution GUI interface.

4.5.2 What needs to be accomplished

- Improve the response of the manual STA movement.
- Improve the advisory display – decide on what and when advisories are displayed and implement the decisions.
- Start design to deal with provisional advisories and semi-automatic functionality.

Appendix A: CTAS processes and hardware configurations

CTAS process and hardware configuration for the meet-time validation and performance testing												
trial date	trial no.	trail type	input file	No. of AC	cm	ra	dp	pfs_c	pgui	ism	sim_man	pilot_man
25-Feb-02	1-7	validation	-	1	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
26-Feb-02	1-4	validation	-	1	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
26-Feb-02	5, 6	validation	-	10	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
27-Feb-02	1, 2	validation	-	10	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
27-Feb-02	4	perf. and validation	-	1	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	heron (333 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
27-Feb-02	5	perf. and validation	-	1	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	heron (333 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
27-Feb-02	6	perf. and validation	-	1	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
27-Feb-02	7	perf. and validation	-	1	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	paoc (360 MHz)	heron (333 MHz)	heron (333 MHz)
28-Feb-02	1	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	-	-	-
28-Feb-02	2	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	-	-	-
28-Feb-02	3	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	heron (333 MHz)	egret (333 Mhz)	-	-	-
28-Feb-02	4	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz)	egret (333 Mhz)	heron (333 MHz)	egret (333 Mhz)	-	-	-
1-Mar-02	1	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz), heron (333 MHz), flygirl (440 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	-	-	-
1-Mar-02	2	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz), heron (333 MHz), flygirl (440 MHz)	egret (333 Mhz)	dove (440 MHz)	egret (333 Mhz)	-	-	-
1-Mar-02	3	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz), dove (440 MHz), flygirl (440 MHz)	egret (333 Mhz)	heron (333 MHz)	egret (333 Mhz)	-	-	-
1-Mar-02	4	performance	rush1_14_02.cm_siim	N	paoc (360 MHz)	isle-royale (333 MHz), dove (440 MHz), flygirl (440 MHz)	egret (333 Mhz)	heron (333 MHz)	egret (333 Mhz)	-	-	-

Appendix B: Summary of single aircraft validation results

Summary of CTAS/EDA Single AC Validation Results from 25, 26, 27 February, 2002														
MF: KARLA														
setup items								**input items**			**Results**			
trial date	trial no.	mode	No. of AC	init distance from MF (nm)	Pre-TOD advised descent mach/CAS	pre-TOD STA (A)	TOD advised mach/CAS (kts)	input mach/CAS	changed STA (A)	STA changed by (min:sec)	MF Crossing time (B)	crossing time delta |A-B| (min:sec)	MF crossing alt (feet)	
26-Feb-02	2	descent	1	114	0.72/300	54:38	0.72/300	0.72/300	no	0:00	55:02	0:24 late	7,100	
26-Feb-02	3	descent	1	114	0.72/280	38:38	0.72/280	0.72/280	39:00	-0:22	39:12	0:12 late	8,400	
27-Feb-02	6	descent	1	114	0.72/280	20:38	0.72/280	0.72/280	21:00	-0:22	20:45	0:15 early	8,700	
26-Feb-02	4	descent	1	114	0.72/280	2:38	0.72/280	0.72/280	4:00	-1:22	3:08	0:52 early	8,500	
25-Feb-02	6	descent	1	87	0.72/280	53:24	0.72/220	0.72/220	56:00	-2:36	55:08	0:52 early	15,700	
26-Feb-02	1	descent	1	114	0.72/280	27:12	0.72/220	0.72/220	30:00	-2:48	29:08	0:52 early	14,600	
27-Feb-02	4	descent	1	114	0.72/280	31:01	0.72/220	0.72/220	34:00	-2:59	32:46	1:14 early	14,700	
25-Feb-02	7	descent	1	87	0.72/280	30:50	0.72/220	0.72/220	34:00	-3:10	33:00	1:00 early	13,900	
25-Feb-02	3	cruise	1	262	0.72/280	56:17	0.72/280	0.72/280	57:00	-0:43	57:20	0:20 after	10,100	
25-Feb-02	5	cruise	1	212	0.68/280	20:32	0.68/280	0.68/280	22:00	-1:28	22:00	0:00 after	7,600	
25-Feb-02	4	cruise	1	187	0.72/280	42:18	0.68/280	0.68/280	43:00	-0:42	43:00	0:00 after	7,100	
25-Feb-02	1	cruise	1	162	0.72/280	23:35	0.72/250	0.72/280	no	0:00	23:35	0:00 after	8,600	
27-Feb-02	5	cruise	1	114	0.72/280	55:30	0.72/280	0.72/280	00:00	-4:30	59:55	0:05 early	9,100	
27-Feb-02	7	cruise	1	114	0.72/280	45:00	0.72/280	0.72/280	46:00	-1:00	46:00	0:00	7,200	

Appendix C: Summary stream-class validation results

Summary of CTAS/EDA Stream-Class Validation Results from 26, 27 February, 2002													
MF: KARLA													
setup items								**input items**			**Results**		
trial date	trial no.	mode	No. of AC	init distance from MF (nm)	ACID	Pre-TOD advised cruise mach/CAS	Pre-TOD advised descent mach/CAS	pre-TOD STA (A)	input mach/CAS	changed STA	STA changed by (min:sec)	MF Crossing time (B)	crossing time delta |A-B| (min:sec)
26-Feb-02	5	descent	10	114	AAL1641		0.72/340	17:38	0.72/340		0:00	17:50	0:12 late
					DAL1682		0.72/300	27:38	0.72/300		0:00	27:51	0:13 late
					AAL820		0.72/300	29:38	0.72/300		0:00	29:52	0:14 late
					EGF664		0.72/280	31:38	0.72/280		0:00	31:52	0:14 late
					N969SS		0.72/280	35:38	0.72/280		0:00	35:50	0:12 late
					AAL751		0.72/280	37:38	0.72/280		0:00	37:50	0:12 late
					DAL1918		0.72/280	39:38	0.72/280		0:00	38:40	0:02 late
					AAL1686		0.72/280	41:38	0.72/280		0:00	X	X
					DAL352		0.72/280	43:38	0.72/280		0:00	X	X
					AAL439		0.72/275	45:38	0.72/275		0:00	45:38	0:00
26-Feb-02	6	descent	10	114	AAL1641		0.72/300	7:38	0.72/300		0:00	7:50	0:12 late
					DAL1682		0.72/240	8:39	0.72/240	11:00	-2:21	10:14	0:46 early
					AAL820		0.72/300	10:38	0.72/300		0:00	10:50	0:12 late
					EGF664		0.72/280	12:38	0.72/280		0:00	13:00	0:12 late
					N969SS		0.72/280	16:38	0.72/280		0:00	16:52	0:14 late
					AAL751		0.72/280	18:40	0.72/280		0:00	19:00	0:20 late
					DAL1918		0.72/280	20:38	0.72/280		0:00	20:50	0:12 late
					AAL1686		0.72/280	22:38	0.72/280		0:00	22:37	0:01 early
					DAL352		0.72/280	24:38	0.72/280		0:00	24:34	0:04 early
					AAL439		0.72/340	17:39	0.72/340	26:00	-8:21	25:50	0:10 early
27-Feb-02	1	cruise	10	114	AAL1641	0.72/250	0.82/280	?	0.82/280	30:00	?	30:27	0:27 late
					DAL1682	0.72	0.72/280	31:38	0.72/280		0:00	30:47	0:50 early
					AAL820	/280	??	33:38	0.72/280		0:00	33:30	0:08 early
					EGF664	0.82	0.80/280	34:39	0.80/280		0:00	34:30	0:09 early
					N969SS	0.82	0.80/280	38:38	0.80/280		0:00	38:30	0:08 early
					AAL751	0.72	0.72/280	40:38	0.72/280		0:00	X	X
					DAL1918	0.70	0.72/280	42:38	0.72/280		0:00	42:20	0:18 early
					AAL1686	0.72	0.72/280	44:38	0.72/280		0:00	44:25	0:13 early
					DAL352	0.72	0.72/280	46:38	0.72/280		0:00	46:45	0:07 late
					AAL439	0.72	0.72/280	48:38	0.72/280		0:00	48:38	0:00
27-Feb-02	2	cruise	10	114	AAL1641	0.72	0.72/280	12:38	0.72/280		0:00	12:30	0:08 early
					DAL1682	0.82	0.82/280	13:39	0.82/280		0:00	13:30	0:09 early
					AAL820	0.72	0.72/280	15:38	0.72/280		0:00	15:30	0:08 early
					EGF664	0.72/250	0.68/280	18:39	0.68/280	18:00	+0:39	18:20	0:20 late
					N969SS	0.72	0.72/280	21:38	0.72/280		0:00	21:30	0:08 early
					AAL751	0.72	0.72/280	23:38	0.72/280		0:00	23:30	0:08 early
					DAL1918	0.72/0.82	0.82/280	17:38	0.82/280	25:00	-7:22	25:30	0:30 late
					AAL1686	0.72	0.72/280	27:38	0.72/280		0:00	27:40	0:02 late
					DAL352	??	??	29:38	0.72/280		0:00	29:30	0:08 early
					AAL439	??	??	30:39	0.72/280		0:00	30:30	0:09 early

Appendix D: Summary meet-time performance results

Summary of CTAS/EDA Performance Results from 27, 28 Feb and 1 Mar, 2002									
trial date	trial no.	mode	No. of AC	average duration of MT calculation (sec)	min MT duration (sec)	max MT duration (sec)	number of cycles	number of MT calculations	duration of trial (min)
27-Feb-02	4	descent	1	0.001989	0.000933	0.031798	86	223	17.3
27-Feb-02	5	cruise	1	0.002576	0.001018	0.024194	83	113	16.6
27-Feb-02	6	descent	1	0.000687	0.000148	0.001303	77	173	15.4
27-Feb-02	7	cruise	1	0.001042	0.000727	0.001499	93	113	18.6
28-Feb-02	1	descent	N	0.001660	0.000139	0.028722	73	1491	14.6
28-Feb-02	2	cruise	N	0.002156	0.000159	0.031621	78	1130	16.5
28-Feb-02	3	descent	N	0.003553	0.000185	0.381343	74	1540	16.5
28-Feb-02	4	cruise	N	0.002963	0.000315	0.090659	78	1141	17.2
1-Mar-02	1	descent	N	0.001438	0.000139	0.127335	47	2042	9.4
1-Mar-02	2	cruise	N	0.002593	0.000145	0.002593	54	1521	10.8
1-Mar-02	3	descent	N	0.002652	0.000181	0.282012	48	1497	9.8
1-Mar-02	4	cruise	N	0.002479	0.000183	0.172646	47	1453	9.7

Appendix E: User's Guide

The following describes the steps of running the CTAS - EDA tool to someone who has limited experience with running CTAS.

1. SETUP

The CTAS processes that are required for running the EDA tool with a captured data are CM, RA, DP, PFS_C and PGUI.

For running with a live radar feed, ISM and a radar daemon are also needed.

There may be multiple RA processes, depending on the number of aircraft in the simulation. It is best to distribute the processes over multiple computers. RA and PFS_C cannot be run on the same processor and multiple RA's must run on their own processors.

2. STARTING CTAS

The user will log onto the computers of their choice and (at NASA) set the proper view. Aliases may be provided for the start-up commands; however, presented here are the command line arguments needed for running the EDA tool. Also the DISPLAY environment variable must be set for any processes that have a display and are not running on the machine at which the user is sitting.

Processes	Directory (ctas/realtime_procs)	Command & Notes
CM	comm_mgr	./cm –data ZFW_DFW –add_all_fps –treat_cids_ambiguous Then run with a prerecorded data set
RA	route_analyzer	ra –data ZFW_DFW –host <*cm machine*> If resources allow, there should be multiple RAs running each on their own processor. Spawns a TS process.
DP	dynamic_planner	dp –data ZFW_DFW –host <*cm machine*>
PFS_C	profile_selector_cntr	pfs_c –data ZFW_DFW –host <*cm machine*> -ms_steps 60 -CDY 2 Must not be run on same processor as RA. Spawns a TS process
PGUI	planview_display	pgui –data ZFW_DFW -host <*cm machine*> -eda –cp –alt –tl -default EDA

For running with live data before starting the processes listed above, make sure RADAR_DAEMON is running and run ISM.

Processes	Directory (ctas/realtime_procs)	Command & Notes
ISM	input_source_mgr	ism –data ZFW_DFW

3. INPUT ON CM GUI

Once all of the processes are running, the user selects the data source.

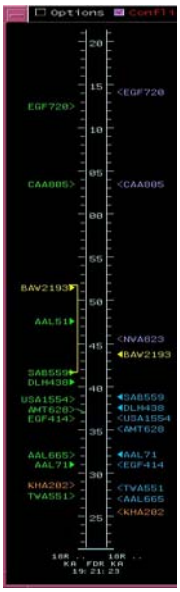
Selection of input file (containing pre-recorded data):
In the second section on the CM GUI, there is a check box labeled "Prerecorded aircraft". Directly below this, there is a label "file:". Enter the path and file name of the desired data file to use as input, into the text box on the right of the "file:" label.

(EXAMPLE data file: /home/annad/CTAS_SCEN/output/KARLA_md80.cm_sim) and then click the "prerecorded aircraft" checkbox. Aircraft (denoted with "&") should now appear on the PGUI. If not, you may need to skip ahead in the data file, by entering an elapsed time into the CM GUI and pushing the SKIP button. To restart the reading of the data file, un-check and re-check the "prerecorded aircraft" checkbox or enter a new data file name and click the NEW button.

Using Live Data for the ZFW center:
In the top section on the CM GUI, there is a check-box labeled ISM. Make sure that the computer listed in the field next to the ISM checkbox is the one on which ISM is running. If it is correct, check the ISM checkbox. If it is not, enter the correct computer name and then check the checkbox. If everything connects, the "FtWorth Ctr:" label should be highlighted. Enter the name of the computer on which the RADAR_DAEMON is running. Normally, at NASA this is "owl". Then check that textbox. After an initialization period, aircraft should then appear on the PGUI display.

4. EDA RELATED GUI INTERACTIONS

The following describes the GUI interactions used with the EDA tool.

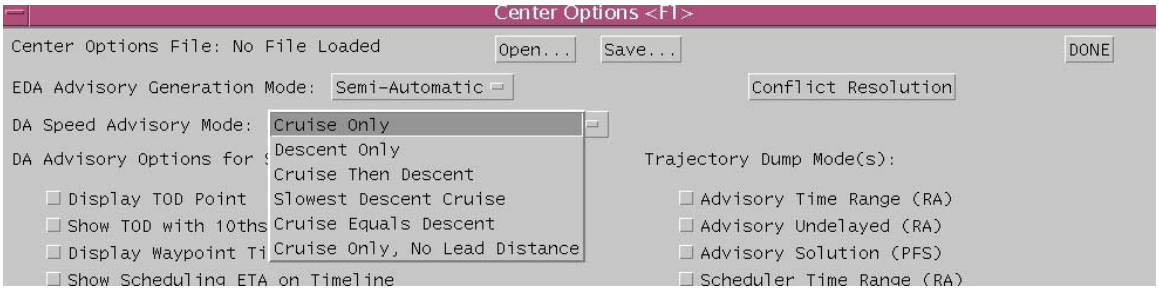


4.1 Timeline Modifications

Typically there is at least one time line displayed on the left of the PGUI, where ETA is on the left (of the timeline) and STA is on the right. This is configurable via the F5 menu from the PGUI.

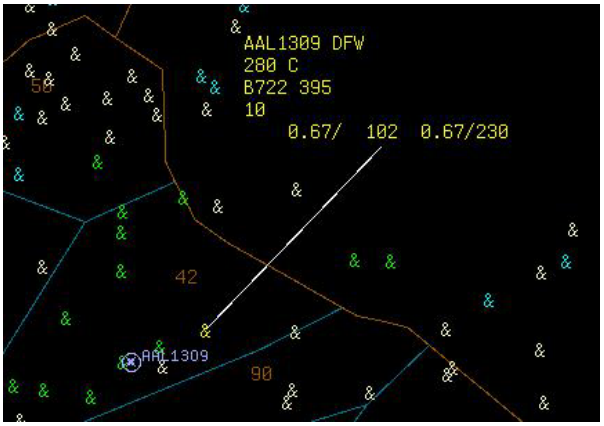
4.2 Speed Mode Modifications

The two speed modes that currently exist are "Descent Only" and Cruise Only". These modes are selected from the F1 menu from PGUI. The default speed mode in EDA is “Descent Only”.



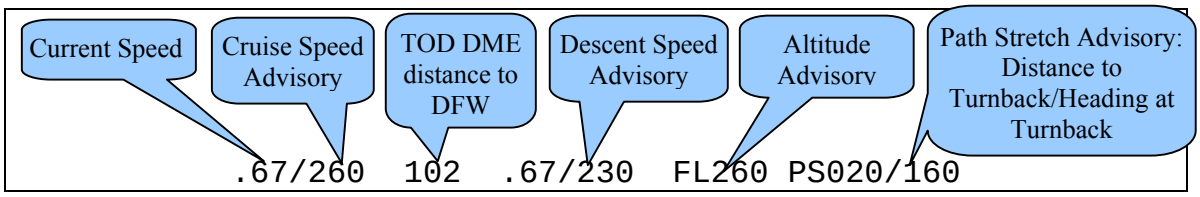
4.3 Aircraft data tag display.

The aircraft data tag is displayed on the PGUI by placing the cursor over the aircraft symbol (&) and clicking the left mouse button, or by dwelling on the aircraft symbol and pressing the “l” (el) key.



4.4 The Advisory in the Data Tag

Cruise speed, distance to top-of-descent (TOD), descent speed, altitude and path stretch advisories may be displayed in the data tag. The format is as follows:

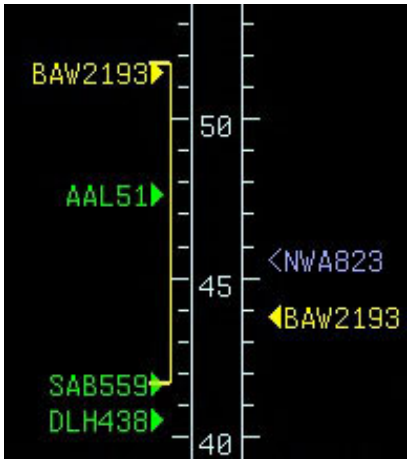


4.5 Menu Bar

Toggle the menu bar (top of PGUI) to remove the advisory display panel.

4.6 Selecting Aircraft and Displaying Time Bracket

The time bracket for an aircraft is displayed on the ETA timeline by placing the cursor over the ACID on the ETA or STA timeline and clicking the left mouse button. This action selects the aircraft and will also display the "ADVISORY DISPLAY". Dwelling on any blank area on the PGUI, hitting the "a" key and entering the ACID also selects the aircraft. The time bracket may be displayed on the STA side of the time line by selecting that option from the <F1> panel.



4.7 Manual Manipulation of STA

After the speed-bracket is displayed, the STA for that ACID may be dragged along the STA timeline using the middle mouse button. The user presses down on the middle mouse button and waits for the ACID to turn white. Once it is white, the user moves the STA to the desired value and releases the button.

If the STA is kept within the speed-bracket (as shown in 4.6 above), then the meet-time algorithms should be able to adjust the trajectory of the aircraft so that the user specified STA is met. If the STA is dragged outside of the speed-bracket, then speed mode alone will not be able to adjust the trajectory to meet the user specified STA.

4.8 Waypoint Display

Waypoints (e.g., dfw, karla, bambe, howdy, fever) can be displayed on the PGUI by:

- pressing the "w" key

- typing in the desired waypoint designator

4.9 Scratch Pad

Placing the cursor on an empty spot on the PGUI and pressing the "s" key can move the "scratch pad". The scratch pad is used for entering information (e.g., waypoint name) and is where various messages are written (e.g., Path Stretch on/off).

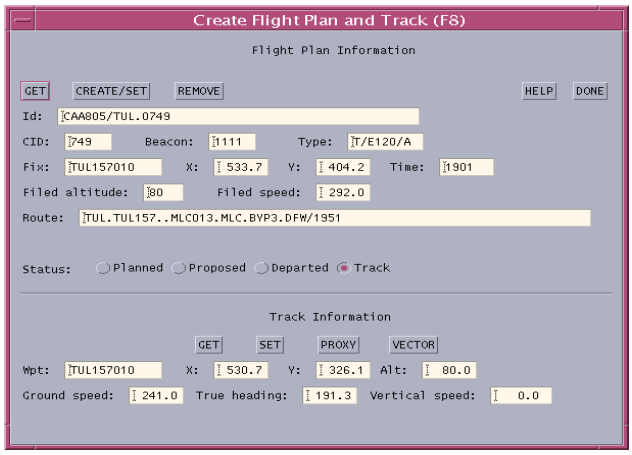
4.10 Route Display

Various data (e.g., flight plan route) can be viewed by selecting it from a pop-up menu for each aircraft. Place the cursor over the aircraft symbol (&) and click the right mouse button. If the aircraft is near a waypoint or another aircraft, a list will appear containing the ACID and the waypoint/aircraft. Left-mouse click on the ACID and a subsequent menu will appear from which one can select various data to be displayed on the PGUI. If the aircraft is not near another object, the subsequent menu will appear directly.

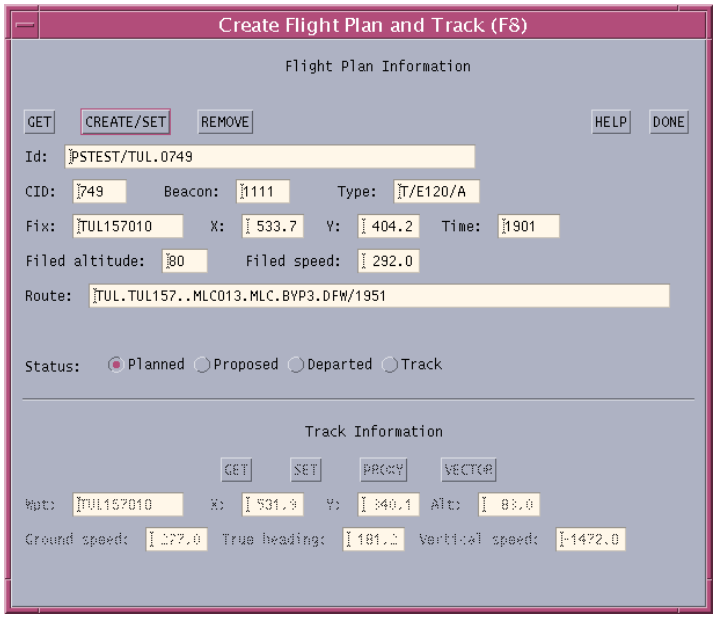
4.11 Path Stretch:

In order to test Path Stretch, a fake aircraft must be created so that it may be controlled. A fake aircraft is created as follows.

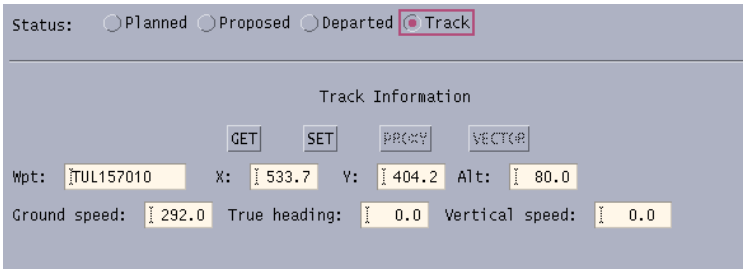
- Bring the CM GUI to the front.
- Press the <F8> key while the cursor is on the CM GUI.
- Enter the call sign of an aircraft arriving at the Feeder Fix that is being displayed on the timeline and press the "GET" button on the upper left side of the panel.



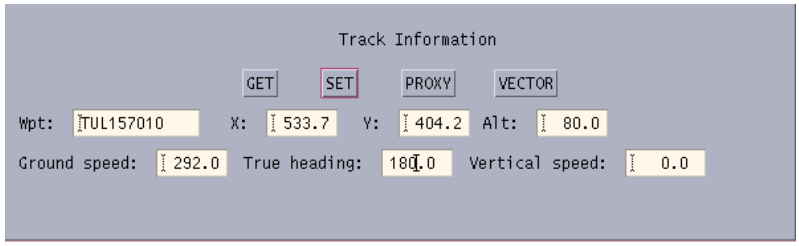
- Change the name of the aircraft to a unique identifier that will be easy to spot and press the "CREATE/SET" button.



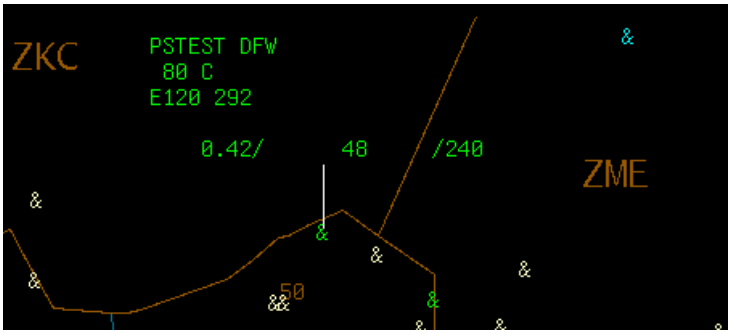
- Press the Track button. This will fill in all fields in the bottom section of the window (representing the fake aircraft) but leave the heading to 0.0.



- Enter a reasonable heading for the fake aircraft and press the "SET" button on the lower panel.

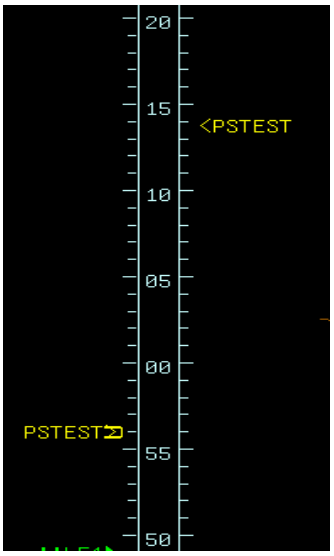


Now go back to the PGUI. The new (fake) aircraft should now show up on the display and on the timeline. Once the fake aircraft has an advisory showing in the data tag we know it is in the system correctly.



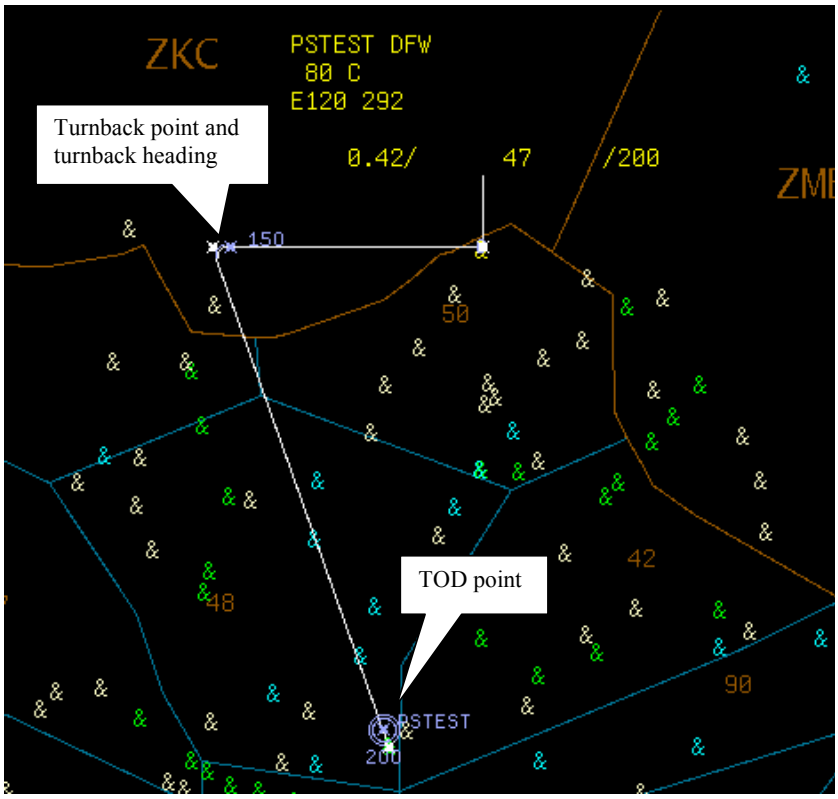
Then initiate the following actions.

- Select the aircraft by left clicking on the aircraft ID on the timeline.
- With the middle mouse button, drag the STA of the fake aircraft out of the advisory time range (ATR), shown by the time bracket, so that it is later than the latest ATR.



- You may turn on the trend vector to follow what the aircraft is doing by dwelling on the aircraft symbol and pressing the "t" key.
- Turn on Path Stretch for this aircraft by dwelling on the aircraft symbol and pressing the "s" key.

Then return to the CM <F8> panel and change the heading enough to initiate the path stretching algorithm. This amount is determined by the "Min Turnout Angle" set on the PGUI <F1> panel. Return to the PGUI and observe the fake aircraft turn. After a short time, a symbol should appear that designates the turnback point and the heading the aircraft should take to capture the default waypoint. Selecting the profile selector route to be displayed for the fake aircraft, as shown in 4.10 Route Display above, will draw a line from the aircraft's current position through the turnback point to the capture waypoint.



Termination:
To end a CTAS run; press the QUIT button on the CM GUI. If ISM is running terminate it via CTRL-C in the terminal window in which ISM was started.